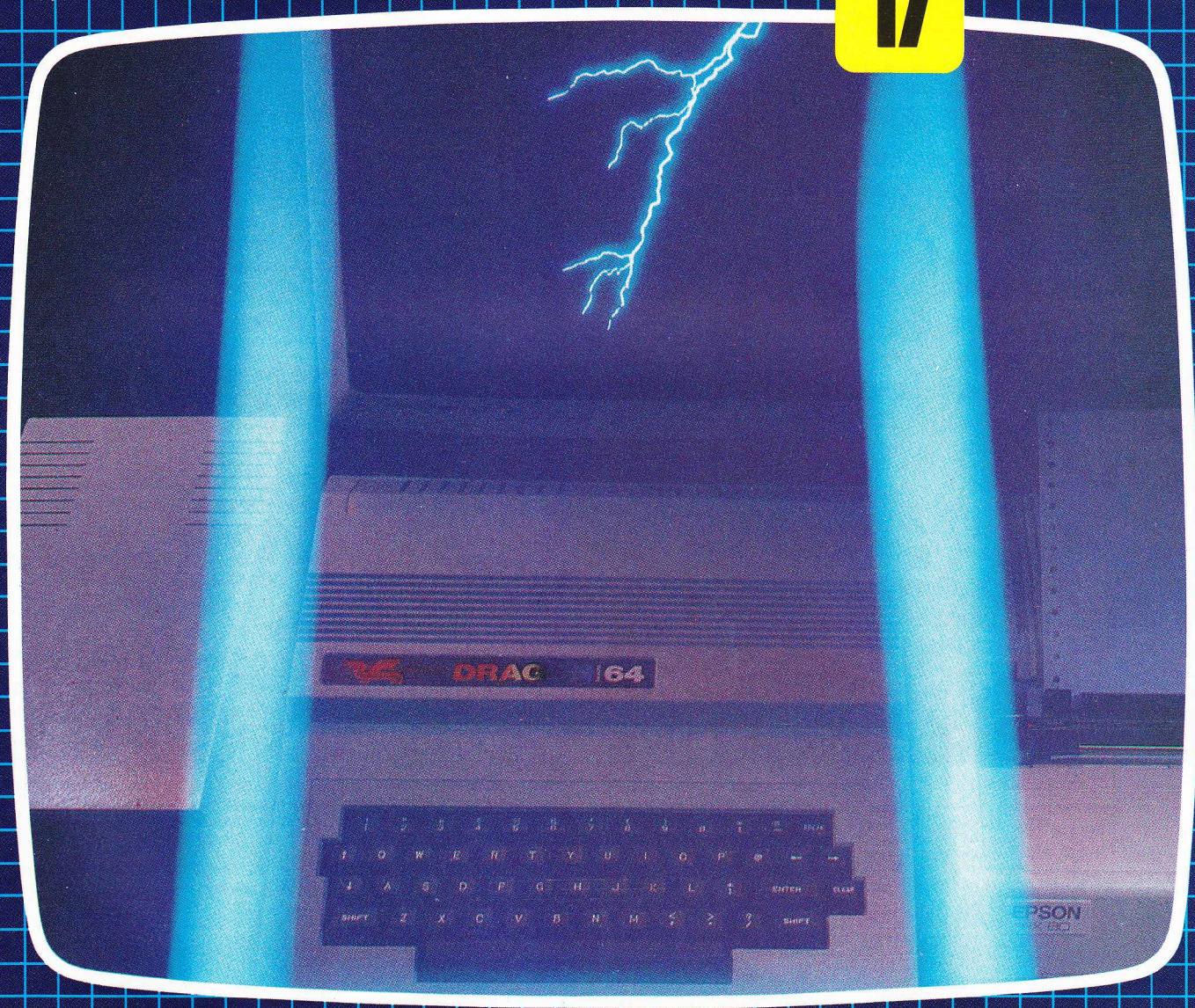
MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

17

Cr$ 10.250



Sistemas operacionais • Epson PX-8
Produtores de melodia eletrônica

rioGráfica

MANAUS, ALTAMIRA, MACAPÁ, JI.PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 13.330

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Produtores de melodia 337

PROJETOS DE PROGRAMAS

Graus de precisão 340

LINGUAGENS

Dupla dinâmica 342

RAIO X

Epson PX-8 345

SOFTWARE

TK!Solver 348

Operador discreto 350

CHIPS & BYTES

História do microprocessador 353

PERFIL

Polimax 356

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) Qual o nome da interface, hoje popular, que permite a um micro controlar um sintetizador musical?
2) Qual a diferença, em BASIC, entre as variáveis A! e A#?
3) Como se chama o software básico que efetua a comunicação entre o sistema operacional em disco e os componentes físicos do computador?
4) O que se quer dizer com "sistema transparente"?

RESPOSTAS DO TESTE ANTERIOR

1) O que distingue um robô de um simples brinquedo é a possibilidade de programarmos seus movimentos.
2) Os programas que auxiliam a elaboração e edição de outros programas chamam-se "utilitários".
3) Apaga-se uma EPROM irradiando uma luz ultravioleta através de sua janela de quartzo, ou reprogramando toda a EPROM com bytes FF (dígitos 1).
4) O visor que produz os dígitos de 0 a 9 em calculadoras, relógios digitais e alguns micros é o visor de sete segmentos (ou barras).

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.as e 4.as trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.as e 4.as capas do fascículo 30); 3.as e 4.as capas dos fascículos de 1 a 29; e guardas.

rioGráfica **Diretoria Executiva:** Oscar Neves; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Gerente de Planejamento:** João Azeredo Coutinho; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Aylton Menezes; **Diretor de Vendas:** Marcos March; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Maurício Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Editor-assistente:** Claudio Marcondes; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Adalberto Cornavaca; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón; **Colaboradores - Editores-assistentes:** Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Carlos Eduardo Silveira Matos; Jussara Maturo; Paulo Brito; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Luiz Roberto de Godoy Vidal; Maria Clara Cescato; Maria Ester Menezes Martinho; Newton Roberval Eichemberg; Regina Amarante; **Pesquisa:** Stella Maria Quentel; **Arte:** Edemilson Ribeiro da Silva; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; Lauro de Lauro; Pierluigi Piazzi; Sérgio Jorge Batarce; Lucas Kenishi Shirahata. As fotos não creditadas pertencem à obra original. Seguem os créditos das demais: © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa. **Composição:** Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Lda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

PRODUTORES DE MELODIA

Processados em bancos de filtros sob o controle de um micro, ruídos sem qualquer atrativo ganham ressonância e modulação, transformando-se em sons inauditos — os elementos da música do futuro.

O parentesco entre a música e a matemática vem de longe. Já no século VI a.C., o filósofo grego Pitágoras ficou intrigado ao ouvir, numa oficina de ferreiros, os martelos soarem tão "afinados" quando golpeavam as bigornas. Pesando vários deles, descobriu que um martelo com metade do peso de outro produzia um som com exatamente o dobro da freqüência — uma oitava acima — produzida por este. Estava descoberta a primeira das leis que governam as relações entre as alturas dos sons.

Na Idade Média, as catedrais ressoavam com os acordes de missas e motetos (composições corais polifônicas), cujo ritmo era tão preciso quanto suas próprias linhas arquitetônicas.

Muitas vezes, essa estrutura melódica alcançava tal complexidade que, segundo se acreditava, apenas os ouvidos de Deus podiam apreciar as relações numéricas contidas nas partituras, enquanto os simples mortais somente ouviam a música. Em vista desse caráter matemático, nada mais natural que o encontro dos universos da música e da computação.

Um desenvolvimento tecnológico recente que vem atraindo muita atenção é a MIDI (Musical Instrument Digital Interface, "interface digital para instrumentos musicais"), dispositivo que permite a intercomunicação de instrumentos eletrônicos (sintetizadores, computadores rítmicos e digital delays) e seu controle por micros pessoais ou dedicados (digital sequencers). Como a maioria dos instrumentos desse tipo hoje produzidos segue o protocolo MIDI, um novo horizonte se abre aos usuários de micros entusiastas de música.

A MIDI, contudo, não faz milagres. Não transformará, da noite para o dia, um operador de micro num Vivaldi ou num Egberto Gismonti. Os resultados sempre dependerão da habilidade musical e da imaginação de cada um, seja a música executada num sistema de sintetizadores ou num violão.

A música eletrônica é mais antiga do que se imagina. Bem antes da Segunda Guerra Mundial (1939-1945) alguns músicos já faziam experiências com geradores de ondas senoidais. Tais dispositivos elétricos provocavam vibração numa lâmina metálica, produzindo um som constante, cuja altura podia ser variada. Esse efeito sonoro foi muito usado nos filmes de ficção científica na década de 50. Por outro lado, os primeiros órgãos Hammond, lançados por volta de 1930, eram eletromecânicos e produziam vários tipos de timbres por meio da síntese aditiva — isto é, da sobreposição de senóides, gerando formas de ondas mais complexas.

Foi o rápido desenvolvimento da eletrônica durante a Segunda Guerra Mundial — em especial o surgimento da gravação magnética, na Alemanha — que proporcionou aos músicos ferramentas para a criação e manipulação dos sons de forma inédita. Eles começaram encomendando fitas com sons previamente gravados, musicais ou não. Esses minúsculos fragmentos

Partitura eletrônica
A "nova música" exigiu uma notação musical diferenciada. Esta partitura, por exemplo, do compositor alemão Karlheinz Stockhausen, foi elaborada de modo a lembrar circuitos eletrônicos. Ela contém representações de sons, compassos, sincronizações e até dá a posição dos microfones.

Impulso inicial

A música eletrônica não é tão nova quanto parece. O compositor britânico Ron Grainer *(sentado)* foi autor da primeira trilha eletrônica a conseguir algum sucesso — o tema do seriado *Doctor Who*, transmitido pela BBC para toda a Europa, no início dos anos 60.

de fitas eram então meticulosamente combinados de modo a formar seqüências de eventos sonoros. A "nova música", como veio a ser conhecida, rompeu todas as regras da teoria musical ortodoxa e, ao mesmo tempo que fascinava alguns ouvintes, torturava muitos outros.

Os dispositivos para variar e distorcer os sons elementares produzidos nos primeiros osciladores, bem como para filtrar e modelar o resultado, foram se tornando mais controláveis, além de disponíveis em maior quantidade. Durante a década de 50, compositores como Karlheinz Stockhausen, na Alemanha, se empenhavam nos pequenos estúdios das emissoras de rádio, criando música eletrônica "pura". Em Paris, trabalhando com engenheiros de som da ORTF (Office de Radiodiffusion et Télévision Française), Pierre Schaeffer foi o pioneiro no que batizou de "música concreta", uma montagem musical utilizando sons do dia-a-dia.

Nos Estados Unidos, a Bell Telephone Laboratories construiu aquilo que veio a ser, provavelmente, o primeiro sintetizador. Ocupava várias salas e tinha por finalidade básica o estudo da síntese da voz humana. A empresa sabia que suas telefonistas com freqüência se enganavam por causa dos diferentes sotaques dos usuários. Isso resultava em elevado número de ligações erradas. Os especialistas — talvez muito otimistas para a época — esperavam resolver o problema com um sistema de respostas automático que usasse um sintetizador de voz universal.

Nessa época, graças à própria Bell, vários músicos americanos receberam seu batismo na eletrônica. Na Grã-Bretanha, trabalho similar estava sendo feito, se bem que em escala menos ambiciosa. Ainda assim, a BBC Radiophonic Workshop produziu um dos clássicos da música eletrônica, no início da década de 60 — o tema musical do seriado de televisão *Doctor Who*.

Software musical

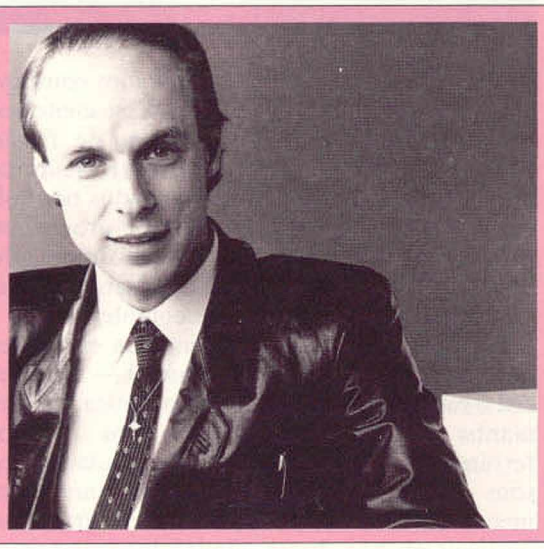
O primeiro encontro entre a música e o computador ocorreu em 1957, quando Lejaren Hiller elaborou um programa para o computador Illiac, da Universidade de Illinois. As instruções eram decompostas em quatro agrupamentos de dados e depois transcritas em notação musical. Resultou uma obra para quarteto de cordas, em quatro movimentos, denominada *Illiac Suite*. A música em si, embora bem-arranjada — para cello, viola e dois violinos —, soava tortuosa e vaga. Mas não é difícil encontrarmos peças musicais da mesma época, produzidas convencionalmente, com sonoridade bem pior.

Poucos anos mais tarde, Hiller partiu para outra criação, dessa vez utilizando um computador IBM 7090. Ele fez um programa chamado Musicomp (Music Simulator-Interpreter for Compositional Procedures, "simulador-intérprete musical para procedimentos de composição"), que permitia maior flexibilidade de trabalho. E batizou a obra assim produzida — para um vocalista acompanhado por sons eletrônicos em fita — com o nome de *Cantata para computador*.

Seu trabalho foi apenas parte de um vasto esforço de pesquisas em universidades americanas nos anos seguintes. John Chowning, outro pioneiro, empregou o computador para estudar a percepção do som quando sua fonte muda de lugar. O uso que a Yamaha fez de seus trabalhos de pesquisa tem relação direta com os sintetizadores da série DX, fabricados em meados da década de 80 a partir do princípio de FM (Frequency Modulation, "freqüência modulada").

Com exceção das trilhas sonoras dos filmes de ficção científica, a música eletrônica permaneceu durante vários anos restrita ao campo da música clássica. O público só veio a conhecê-la melhor graças a mudanças de abordagem e de técnica por parte dos compositores de vanguarda. Os concertos dessa nova música, nos anos 60, apresentavam vários intérpretes tocando instrumentos convencionais e outros encarregados de processar eletronicamente estes sons, com unidades de separação de freqüências e filtros.

Desbravadores da música



Brian Eno tornou-se famoso em 1973, quando passou a usar seus sintetizadores em shows e gravações de astros como David Bowie e Robert Fripp. No Brasil, vários músicos, como Egberto Gismonti (o pioneiro), Wagner Tiso e César Camargo Mariano, utilizam sintetizadores em seus trabalhos, juntamente com instrumentos comuns.

Todos podiam estar acompanhando por uma partitura — que pouca semelhança tinha com a notação musical tradicional. Nelas havia, por exemplo, tanto indicações sobre posições de microfones e variações de filtros como anotações visuais relativas ao efeito sonoro a ser produzido. Em alguns casos, os músicos executavam por partituras que mais pareciam gráficos.

Pop eletrônico

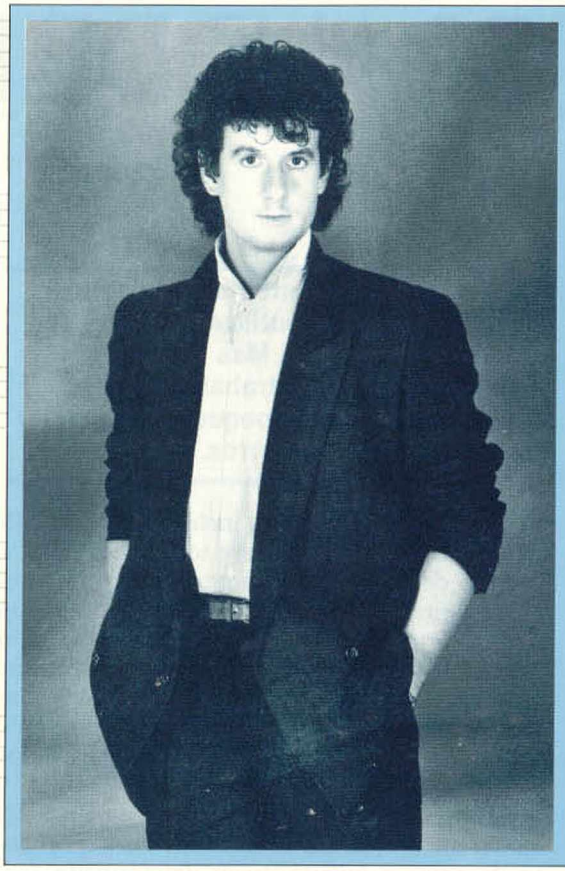
Ainda na década de 60, músicos jovens começavam a passar mais tempo nos estúdios de gravação e a fazer experiências com a música eletrônica. Exemplo são os Beatles, que encontraram em George Martin um perito engenheiro de som e também um músico voltado para os avanços da música clássica de vanguarda. Ele encorajou o conjunto a usar todo o estúdio como instrumento musical, e logo os Beatles estavam trabalhando com técnicas de montagem de fitas e som sintetizado. O sucesso da música eletrônica junto ao grande público ocorreria, contudo, em 1967, com o lançamento da suíte *Switched on Bach,* de Walter Carlos — uma recriação eletrônica das obras de Bach realizada no sintetizador modular MOOG IIIp.

Alguns músicos fizeram nome utilizando determinados dispositivos de processamento de som. O guitarrista Jimmy Page, cujo estilo inspirava-se no dos músicos negros americanos dos anos 40 — só que usando uma série de controles de distorção —, produziu o som característico do Led Zeppelin. Esse som mais o uso que Jimi Hendrix fazia da realimentação e dos filtros de varredura rápida (os "pedais uá-uá") deram origem ao heavy metal.

Na década de 70, as várias unidades geradoras e processadoras de som (disponíveis desde os anos 50) incorporaram transistores em seus projetos. Elas se tornaram menores, leves e, portanto, não precisavam mais se restringir a um estúdio. Os guitarristas podiam fazer música ao vivo, criando inúmeros efeitos sonoros. Logo depois, pianistas e organistas também tiveram acesso aos sintetizadores e puderam levá-los para o palco.

Esses sintetizadores incluíam basicamente um conjunto de osciladores sintonizáveis, modeladores (com os quais se criam as características de irrupção, sustentação e decaimento do som), filtros variáveis, ring modulators (moduladores em anel, capazes de decompor os sinais em novas freqüências) e geradores de ruído.

O equipamento dos estúdios de gravação também se tornava mais sofisticado à medida que os músicos recorriam a eles para obter, no processo de produção, algo que não conseguiam criar no palco. A mesa de mixagem, destinada a combinar gravações em fita magnética de dezesseis ou 24 pistas, era grande demais para ser transportada com facilidade, e muitas das unidades de processamento exigiam longo tempo de montagem. Enquanto isso, nos Estados Unidos e na Grã-Bretanha, estava emergindo uma nova geração de produtores. Não raro, iniciavam como engenheiros e tinham mais familiaridade com o equipamento do que os músicos, de quem recebiam para produzir o "som certo".



Mago da eletrônica

Steve Levine é um consagrado produtor europeu de discos. Seu trabalho ficou conhecido pela excelente estrutura harmônica, que ele obtém combinando vozes, música eletrônica e sons de bateria produzidos pelo sintetizador Linn. Levine desenvolveu técnicas refinadas para os sistemas de gravação digital, criando obras de alta qualidade artística.

Na Jamaica, os engenheiros começaram a usar a mesa de mixagem como instrumento. Músicas inteiras, gravadas em fitas de várias pistas, eram desmontadas. A contribuição vocal e instrumental virou matéria-prima, resultando no pesado estilo dub, dependente da reverberação e das unidades de atraso de sinal.

O advento dos sintetizadores digitais trouxe a possibilidade de codificar sons não eletrônicos — processo conhecido como "amostragem". Os sintetizadores que reproduzem o som das baterias, como o Linn, tornaram-se equipamentos de estúdio muito procurados, logo passando a fazer parte de apresentações ao vivo. Em meados da década de 80, a amostragem e a manipulação de sons representavam a ponta-de-lança da criação musical.

Estúdios modernos e instalações de palco contêm, em geral, mais unidades de equipamento digital do que instrumentos analógicos. Alguns grupos bem-sucedidos combinam suas habilidades musicais às técnicas digitais de produção — como as utilizadas pelo produtor britânico Steve Levine, que trabalha com instrumentos e unidades de processamento de custo milionário.

A necessidade de interface para agrupar todos esses equipamentos e instrumentos (juntando-lhes o sistema operacional e a memória de um microcomputador) acabou reunindo os fabricantes de instrumentos musicais: em abril de 1983, eles apresentaram a primeira especificação da MIDI. A partir daí, poucas empresas ousaram anunciar um sintetizador não compatível com essa interface.

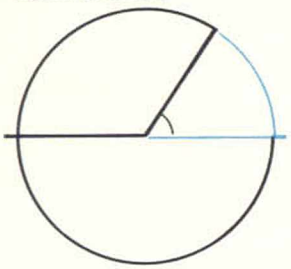
GRAUS DE PRECISÃO

As funções trigonométricas de seno e co-seno podem ser utilizadas nos programas em BASIC. Mas convém testá-las quando se trabalha com números muito pequenos ou muito grandes, para evitar erros.

Como a linguagem BASIC inclui as funções SIN (seno) e COS (co-seno), deve ser fácil calcular a posição de determinado ponto numa linha depois de girá-la certo ângulo. O co-seno de θ dá a posição no eixo x (a coordenada x), e o seno de θ, a posição no eixo y. Mas é preciso lembrar que a maioria das versões do BASIC trabalha com radianos em vez de graus. Assim, o primeiro passo consiste em estabelecer a diferença entre graus e radianos.

Chama-se "arco" qualquer parte de uma circunferência. Quando o arco tem o comprimento do raio, o ângulo assim determinado *(ver figura)* recebe o nome de "radiano". Se o raio for igual a 1, o arco terá o comprimento de uma unidade. Como existem 360 graus numa circunferência e seu comprimento é dado pela fórmula $2\pi r$, facilmente relacionamos graus e radianos:

Medida do radiano



$360° = 2\pi$ radianos;
$180° = \pi$ radianos;
$1° = \pi/180$ radianos $= 0{,}0174$ radiano.

Para achar o co-seno de um ângulo medido em graus, num programa BASIC, devemos primeiro converter a medida para radianos e, em seguida, usar a função COS. Experimente este pequeno programa escrito para o CP 500:

```
10 INPUT "ENTRE O ANGULO EM GRAUS"; A
20 LET B# = A * 0,0174
30 LET C# = COS(B#)
40 PRINT "O CO-SENO DE";A;"GRAUS E";C#
50 END
```

O símbolo # após o nome de variáveis indica que elas armazenam dados com dupla precisão. Empregamos tal notação nas versões do BASIC do TRS-80 e compatíveis — CP 300, CP 500, Digitus, Sysdata etc.

Modificando o programa pela introdução da função seno, podemos construir uma tabela com todos os valores de θ, de 0 a 360 graus. Marcando esses valores no eixo y de um gráfico (o eixo x representa os valores de θ em radianos), o resultado será a curva do seno, o desenho dos pontos de interseção da hipotenusa com a circunferência de raio unitário no eixo y para todos os ângulos de rotação. Trata-se de um modo alternativo de descrever matematicamente uma circunferência.

Talvez a programação em BASIC de seu micro permita que as funções SIN e COS funcionem tanto em graus quanto em radianos, mas em geral isso não acontece. Se você preferir trabalhar com graus, poderá criar uma "função definida pelo usuário" que faça as conversões. Eis um programa que usa esse recurso:

```
10 REM UMA FUNCAO DEFINIDA PARA
   TRABALHAR COM GRAUS
20 DEF FNGRASEN(D#) = SIN(D# * 0,017453293)
30 INPUT "ENTRE O ANGULO EM GRAUS";D#
40 PRINT "O SENO DE";D#;"GRAUS
   E";FNGRASEN(D#)
50 END
```

A linha 20 define a função GRASEN (graus/seno), cujo único parâmetro é a variável de dupla precisão D#. A metade direita da definição mostra como calcular o valor a ser devolvido pela função — o seno de determinado ângulo medido em graus. Para chamar uma função definida pelo usuário, usamos o nome normal da função com o valor que será operado entre parênteses. Observe, no entanto, que se executa a linha contendo a definição antes que se possa chamar a função.

Números, pequenos e grandes

Um dos problemas quanto ao uso da função seno na linguagem BASIC é que nem todas as suas versões conseguem manipular a função corretamente quando o valor do ângulo se aproxima de zero. Nesse caso, o valor do seno de θ também se aproxima de zero: o seno de θ vale zero quando θ é igual a zero.

Em outras palavras, quando o ângulo tende a zero, o mesmo acontece com o arco de circunferência que define θ, enquanto o ponto em que a hipotenusa intercepta o círculo se aproxima cada vez mais de zero no eixo y.

O BASIC é limitado, não conseguindo trabalhar com valores muito grandes ou muito pequenos além de determinados limites. Se θ for muito pequeno — por exemplo, igual a 1,0E-36, ou seja, 1 x 10 elevado à potência (−36) —, o BASIC não conseguirá processá-lo e simplesmente devolverá um valor igual a zero para o seno desse número.

Por esse motivo, antes de usar a função seno, teste o BASIC de seu micro com o pequeno programa que segue, também escrito para o CP 500. O resultado é confiável até que θ atinja um valor menor que 1,0E-38 (37 zeros após a vírgula).

Overflow negativo do seno, ou erros de arredondamento

```
1 REM TESTE PARA FUNCAO SENO
2 REM ERRO DE ARREDONDAMENTO OU ESTOURO NEGATIVO
10 LETX=10/6
20 PRINT"ITERACAO","  VAL DE X","  VAL DE SIN(X)"
30 FOR I=1 TO 20
40 LET X=X/10
50 PRINTI,X,SIN(X)
60 NEXT I
```

```
ITERACAO        VAL DE X          VAL DE SIN(X)
 1               .166667           .165996
 2               .0166667          .0166659
 3               1.66667E-03       1.66665E-03
 4               1.66667E-04       1.66656E-04
 5               1.66667E-05       1.66656E-05
 6               1.66667E-06       1.68529E-06
 7               1.66667E-07       1.87254E-07
 8               1.66667E-08       0
 9               1.66667E-09       0
 10              1.66667E-10       0
 11              1.66667E-11       0
 12              1.66667E-12       0
 13              1.66667E-13       0
 14              1.66667E-14       0
 15              1.66667E-15       0
 16              1.66667E-16       0
 17              1.66667E-17       0
 18              1.66667E-18       0
 19              1.66667E-19       0
 20              1.66667E-20       0
```

Antes de utilizar qualquer operação matemática em BASIC de modo confiável, precisamos conhecer a faixa dos números aptos a serem manipulados com precisão. Também não devemos nos esquecer de que uma variável com um nome constituído apenas de caracteres alfanuméricos, tal como X ou TOTAL, é automaticamente considerada de precisão simples, ou seja, capaz de armazenar no máximo sete dígitos. Mais ainda, algumas variáveis são especificadas como de precisão simples — ou colocadas nessa categoria —, acrescentando-se a elas um ponto de exclamação: X! ou TOTAL!.

As variáveis de precisão dupla, já mencionadas, armazenam dezessete dígitos, sendo especificados pelo sinal #. E as variáveis do tipo integer, que só podem armazenar números inteiros, aparecem denotadas em inúmeras versões do BASIC pelo símbolo %, tal como em X%.

Damos a seguir um pequeno programa para você testar quantos dígitos pode armazenar numa variável em sua versão do BASIC; também é fornecido o resultado da execução no BASIC do CP 500. Existem duas versões do teste, uma para números pequenos e outra para grandes. Na primeira, o programa mostra que, para números inferiores a 3,3 x 10E-38, o BASIC os arredonda para zero. Para números grandes, maiores que 3,3x10E−37, ocorre uma sobrecarga — overflow —, e os resultados deixam de ser confiáveis. Se você tiver de trabalhar com números muito grandes ou muito pequenos, talvez precise escrever rotinas aritméticas especiais para superar essas limitações.

Números pequenos em BASIC

```
1 REM TESTES COM NUMEROS PEQUENOS NO BASIC
10 LET X#=.000033333333333#
20 PRINT"ITERACAO","  PREC DUPLA","  ","  PREC SIMP"
30 PRINT
40 FOR I=1 TO 40
50 LET X#=X#/10
60 LET X'=X#
70 PRINT I,X#TAB(48)X'
80 NEXT I
```

```
ITERACAO        PREC DUPLA                PREC SIMP
 1               3.333333333D-06           3.33333E-06
 2               3.333333333D-07           3.33333E-07
 3               3.333333333D-08           3.33333E-08
 4               3.333333333D-09           3.33333E-09
 5               3.333333333D-10           3.33333E-10
 6               3.333333333D-11           3.33333E-11
 7               3.333333333D-12           3.33333E-12
 8               3.333333333D-13           3.33333E-13
 9               3.333333333D-14           3.33333E-14
 10              3.333333333D-15           3.33333E-15
 11              3.333333333D-16           3.33333E-16
 12              3.333333333D-17           3.33333E-17
 13              3.333333333D-18           3.33333E-18
 14              3.333333333D-19           3.33333E-19
 15              3.333333333D-20           3.33333E-20
 16              3.333333333D-21           3.33333E-21
 17              3.333333333D-22           3.33333E-22
 18              3.333333333D-23           3.33333E-23
 19              3.333333333D-24           3.33333E-24
 20              3.333333333D-25           3.33333E-25
 21              3.333333333D-26           3.33333E-26
 22              3.333333333D-27           3.33333E-27
 23              3.333333333D-28           3.33333E-28
 24              3.333333333D-29           3.33333E-29
 25              3.333333333D-30           3.33333E-30
 26              3.333333333D-31           3.33333E-31
 27              3.333333333D-32           3.33333E-32
 28              3.333333333D-33           3.33333E-33
 29              3.333333333D-34           3.33333E-34
 30              3.333333333D-35           3.33333E-35
 31              3.333333333D-36           3.33333E-36
 32              3.333333333D-37           3.33333E-37
 33              3.333333333D-38           3.33334E-38
 34              0                         0
 35              0                         0
 36              0                         0
 37              0                         0
 38              0                         0
 39              0                         0
 40              0                         0
```

Números grandes em BASIC

```
1 REM TESTES COM NUMEROS GRANDES NO BASIC
10 LET X#=3.333333333333334#
20 PRINT"ITERACAO","  PREC DUPLA","  ","  PREC SIMP"
30 PRINT
40 FOR I=1 TO 40
50 LET X#=X#*10
60 LET X'=X#
70 PRINT I,X#TAB(48)X'
80 NEXT I
```

```
ITERACAO        PREC DUPLA                PREC SIMP
 1               33.33333333333334         33.3333
 2               333.3333333333334         333.333
 3               3333.333333333334         3333.33
 4               33333.33333333334         33333.3
 5               333333.3333333334         333333
 6               3333333.333333334         3.33333E+06
 7               33333333.33333334         3.33333E+07
 8               333333333.3333334         3.33333E+08
 9               3333333333.333334         3.33333E+09
 10              33333333333.33334         3.33333E+10
 11              333333333333.3334         3.33333E+11
 12              3333333333333.334         3.33333E+12
 13              33333333333333.34         3.33333E+13
 14              333333333333333.4         3.33333E+14
 15              3333333333333334          3.33333E+15
 16              3.333333333333334D+16     3.33333E+16
 17              3.333333333333334D+17     3.33333E+17
 18              3.333333333333334D+18     3.33333E+18
 19              3.333333333333334D+19     3.33333E+19
 20              3.333333333333334D+20     3.33333E+20
 21              3.333333333333334D+21     3.33333E+21
 22              3.333333333333734D-22     3.33333E+22
 23              3.333333333333334D+23     3.33333E+23
 24              3.333333333333334D+24     3.33333E+24
 25              3.333333333333334D+25     3.33333E+25
 26              3.333333333333334D+26     3.33333E+26
 27              3.333333333333334D+27     3.33333E+27
 28              3.333333333333334D+28     3.33333E+28
 29              3.333333333333334D+29     3.33333E+29
 30              3.333333333333334D+30     3.33333E+30
 31              3.333333333333334D+31     3.33333E+31
 32              3.333333333333334D+32     3.33333E+32
 33              3.333333333333334D+33     3.33333E+33
 34              3.333333333333334D+34     3.33333E+34
 35              3.333333333333334D+35     3.33333E+35
 36              3.333333333333334D+36     3.33333E+36
 37              3.333333333333334D+37     3.33333E+37
OVERFLOW
```

DUPLA DINÂMICA

Abandonando as limitações até agora impostas a nossos arquivos, veremos conceitos como desalocação, indicadores, stacks e heaps, que permitem o processamento de arquivos de tamanho não especificado.

Dois procedimentos padronizados (NEW e DISPOSE) e um tipo especial de dados (POINTER) proporcionam ao PASCAL uma extraordinária flexibilidade na "alocação dinâmica" e na desalocação de memória. Um tipo POINTER é apenas uma variável que, em vez de conter um dado como seu valor — por exemplo, um valor integer —, indica ou "aponta" um valor integer ou qualquer objeto de dados, estruturado ou não. Para definir um tipo POINTER, usamos uma seta apontada para cima (↑) ou o sinal de acento circunflexo ^ antes do identificador (ou do tipo de dado que desejamos indicar):

```
TYPE
  MATRIZ = ARRAY[1 .. 100] OF REAL;
  INDIRETO = ^MATRIZ;
```

Todos os compiladores PASCAL com padrão ISO também aceitam o símbolo alternativo @. A definição do tipo reserva uma única posição de memória, que será usada para conter o endereço de uma matriz somente após ter ela sido criada pelo procedimento NEW. Enquanto isso, o valor de POINTER permanece indefinido, como ocorreria com qualquer outra variável, de modo que

```
VAR
  ENDERECO : INDIRETO;
  NUMERO : INTEGER;
```

reservará espaço para um endereço na máquina (16 bits num micro de 8 bits) e para um valor integer, sem que sejam inicializados com qualquer valor específico.

Da mesma forma que poderíamos inicializar o valor integer em zero antes de usá-lo, o indicador pode receber o valor especial NIL. Essa palavra reservada do PASCAL significa apenas que o POINTER não está apontando nada, equivalendo ao valor numérico zero (que implica ausência de um número efetivo). As duas variáveis do exemplo receberiam a seguinte atribuição:

```
ENDERECO := NIL;
NUMERO := 0;
```

Exemplificando o emprego de NEW e DISPOSE, usaremos, para simplificar, somente valores integer. Quando se faz necessária uma variável apontadora para indicar um novo item, chamamos o procedimento NEW, que reserva espaço para o item e coloca seu endereço no POINTER. A seguir, o item de dados é referenciado por meio da "referenciação" ao POINTER (P) com a notação P↑ ou P^. Observe que os sinais agora vêm após o identificador do POINTER, podendo ser considerados como "os itens para os quais p está apontando". Não faz sentido referenciar um POINTER indefinido ou que não aponte para algo.

Desalocação dinâmica

Após utilizar os dados criados pelo uso do procedimento NEW, chamamos o procedimento DISPOSE. Trata-se da operação oposta a NEW, porque, com DISPOSE, a memória volta ao conjunto dinâmico, e o valor do POINTER se torna indefinido. O programa DOISMAISDOIS ilustra esses pontos:

```
PROGRAM DOISMAISDOIS (OUTPUT);
TYPE
    POINTER = ^INTEGER;
VAR
    P1,
    P2^ : POINTER;
    RESPOSTA : INTEGER;
BEGIN
  NEW (P1);
  P1^ := 2;
  NEW (P2);
  P2^ := P1^;
  RESPOSTA := P1^ + P2^;
  WRITELN (P1^, ' + ', P2^, ' = ', RESPOSTA);
  DISPOSE (P1);
  DISPOSE (P2);
END.
```

Com esse método pouco usual de adição, notamos o seguinte:

- As variáveis dinâmicas são anônimas — não há identificador de variável (como N) que comporte o valor 2 em qualquer ponto do programa; a referência a esses itens se faz indiretamente.

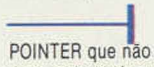
- Após o segundo DISPOSE, a única memória usada é um valor integer isolado (RESPOSTA); todos os dados dinâmicos deixaram de existir.

O endereçamento indireto implícito no uso do POINTER tem duas conseqüências. Em primeiro lugar, nunca sabemos (nem precisamos saber) quais os endereços efetivos. O único "endereço absoluto" disponível para o programador é NIL.

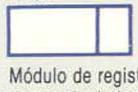
Do mesmo modo, estamos livres para usar, corrigir e, em seguida, reutilizar qualquer parte da memória disponível sem que seja preciso reorganizar os dados descartados. O PASCAL cuida do gerenciamento da memória, e a única informação de que precisaremos refere-se à quantidade de memória disponível. Para obtê-la, empregamos a função não padronizada MEMAVAIL ou, em alguns equipamentos, a função FREE, que fornece o número restante de bytes de memória.

Símbolos indicadores

POINTER indefinido (após uma declaração, antes de uma atribuição ou após DISPOSE).



POINTER que não aponta para nada (após atribuição NIL).



Módulo de registro com um campo de dados e um POINTER.

Outra extensão útil é a função SIZEOF (TYPEIDENTIFICADOR), que fornece o tamanho em bytes de um determinado tipo. A RAM do usuário se divide em estruturas internas — o "stack" e o "heap". Emprega-se o primeiro para chamadas a funções e procedimentos, armazenando seus endereços de retorno, dados locais e valores resultantes. Todos os dados dinâmicos são alocados no heap, que opera de forma semelhante ao stack. No entanto, o heap não é uma estrutura de dados LIFO (Last In First Out, "último a entrar, primeiro a sair") e inicia no lado oposto da memória, crescendo em direção ao stack. O uso ambicioso de NEW e/ou da recorrência pode levar a uma "colisão stack-heap", evitável por meio de

```
IF SIZEOF (OBJETO) > PORCENT * MEMAVAIL
  THEN ..
```

sendo OBJETO o identificador de tipo dos dados que estão para ser criados e PORCENT um valor na região de 0,7, que reserva 30% da memória para o stack e 70% para o heap. Se — como é efetivamente o caso — MEMAVAIL/FREE funcionar como uma "marca de limite máximo", será possível manter uma "contagem de dados não em uso" dos itens descartados.

Estruturas ligadas

Percebemos a verdadeira força dos POINTERS quando criamos estruturas ligadas (linked) em forma de árvore, listas vinculadas individual ou duplamente, estruturas circulares etc. No caso de strings de caracteres, podemos — e freqüentemente o fazemos — utilizar uma matriz com um tamanho fixo de, digamos, oitenta caracteres. Se tivermos uma matriz de strings para armazenar um documento, por exemplo, cada linha em branco ainda ocupará 80 bytes de armazenamento. Do mesmo modo, não podemos representar linhas com mais de oitenta caracteres — toda a estrutura de dados é rígida demais.

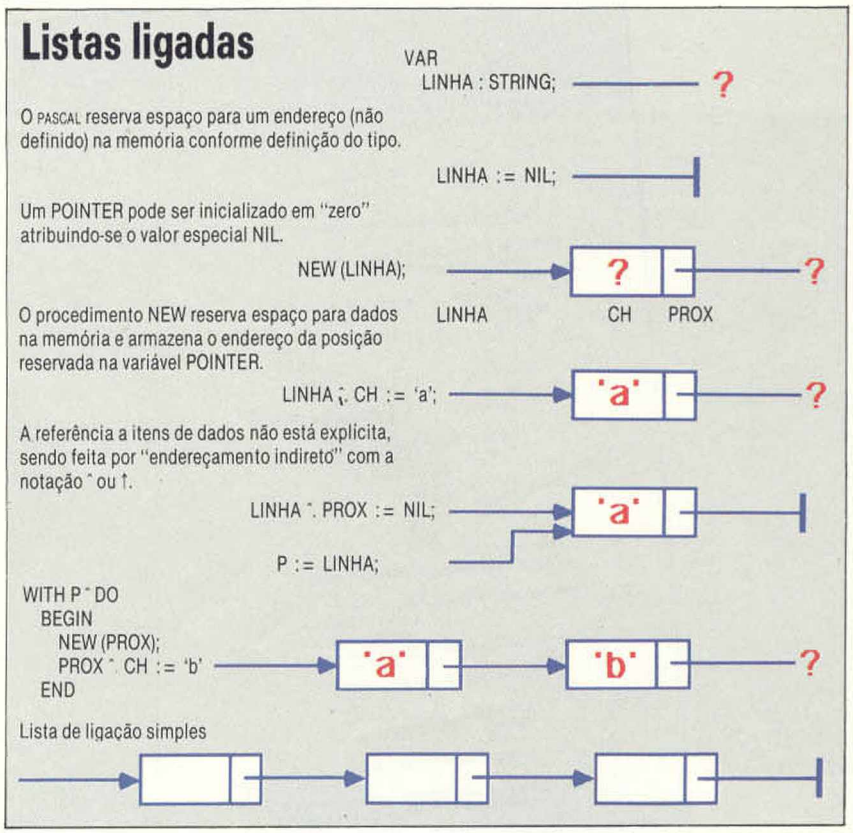
Um compilador PASCAL tem de distinguir entre identificadores de qualquer tamanho. Assim, por exemplo, como representar com precisão a variação de tamanho característica do mundo real? A estrutura mais natural seria um registro contendo dois campos: um para cada item de dados (CHAR, nesse caso), e um segundo campo POINTER apontando o registro seguinte, se houver, na lista:

```
TYPE
    STRING = ^CARACTERE;
    CARACTERE = RECORD
                  CH : CHAR;
                  PROX : STRING
                  END;
VAR
    LINHA : STRING;
BEGIN
  LINHA := NIL;
  ...
```

Representa-se um string vazio pela atribuição do valor NIL. Qualquer outra seqüência exigirá um novo registro contendo cada CHAR e um outro POINTER para indicar o registro seguinte. O último registro terá seu próximo campo inicializado em NIL para que se possa detectar o final do string. Um procedimento para imprimir o string seria:

```
WHILE LINHA <> NIL DO
  BEGIN
    WRITE (LINHA ^. CH);
    LINHA := LINHA ^. PROX
  END
```

Observe a recorrência da estrutura, definida em termos de si mesma. Não se pode atribuir ao campo indicador um tipo que tenha sido definido por completo; assim, faz-se uma referência "antecipada".



De modo sucessivo
O PASCAL permite ao programador criar "listas ligadas" — poderosas estruturas de dados nas quais cada item aponta o item seguinte da lista. Esta pode ser de ligação simples (entre itens sucessivos), de ligação dupla (cada elemento possui POINTERS tanto para o item seguinte como para o anterior) ou circular (onde o último item, numa lista de ligação simples, aponta o primeiro item).

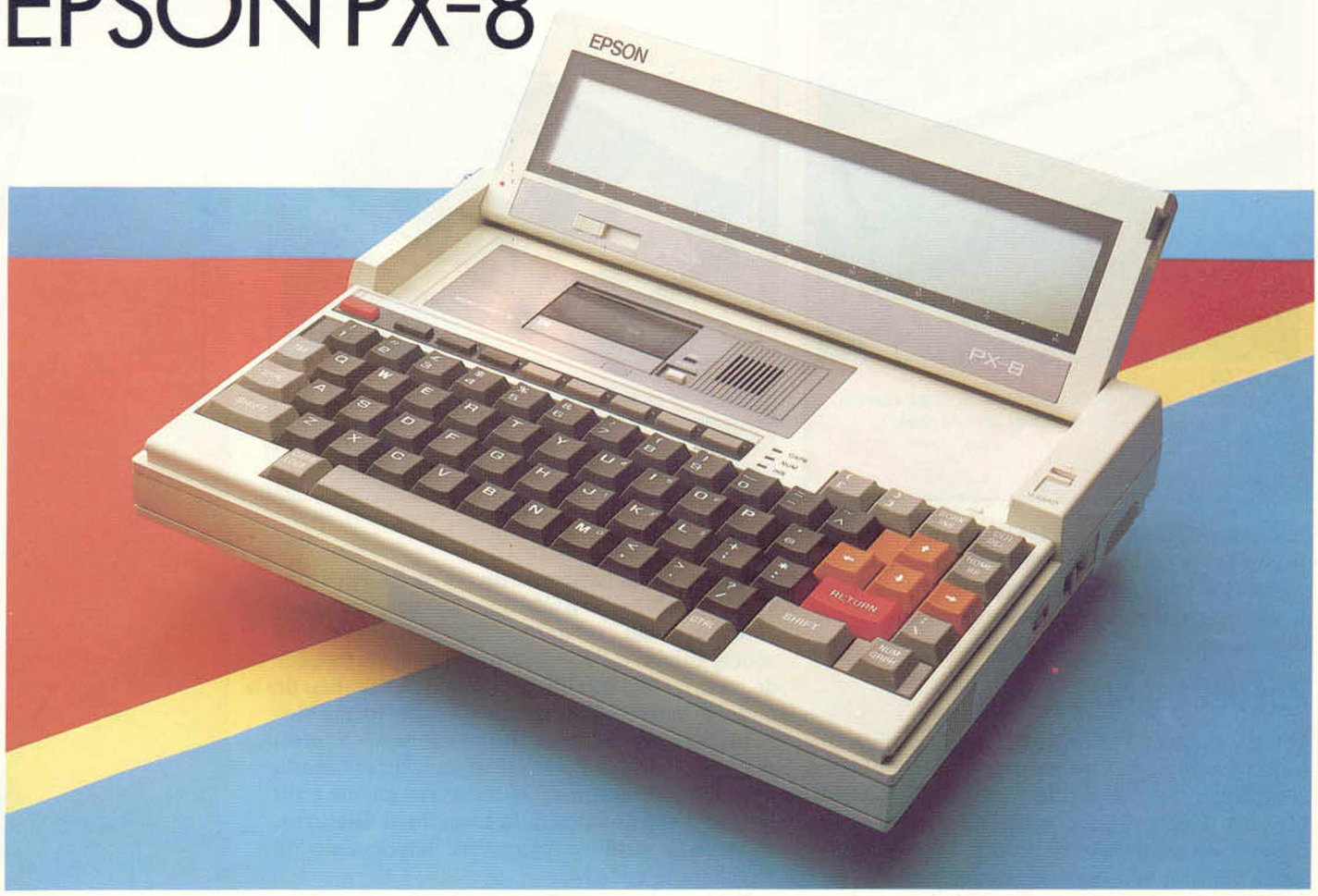
Programa ListaCirc

```
PROGRAM LISTACIRC (INPUT, OUTPUT);
        {PROPOSITO: INSERIR REGISTROS CONTENDO DADOS
          DE VARIOS TIPOS EM UMA LISTA CIRCULAR
          DINAMICAMENTE ALOCADA. OS DADOS SAO
          COLOCADOS EM ORDEM DE UM CAMPO-CHAVE
          (NOME) ALFABETICAMENTE, TORNANDO
          DESNECESSARIOS ALGORITMOS DE CLASSIFICACAO}

CONST
    MAXSTRING  =  25;
    ESPACO     =  ' ';

TYPE
    CARDINAL   =  0 .. MAXINT;
    TAMSTRING  =  0 .. MAXSTRING;
    STRING     =  PACKED ARRAY [TAMSTRING] OF CHAR;
    REGISTRO   =  RECORD
                    NAME   : STRING;
                    {OUTROS CAMPOS}
                    DEBITO : CARDINAL
                  END;

    NO         =  RECORD
                    ITEM   : REGISTRO;
                    PROXIMO : POINTER
                  END;
```

Linha circular
Este programa permite inserir registros contendo dados mistos em uma lista circular dinamicamente alocada.
Os dados são organizados por meio de uma chave alfabética (NOME), tornando assim desnecessários os algoritmos de classificação.

```
    POINTER        =     NO
  VAR
    LISTA                  :  POINTER;

  |1111111111111111111111111111111111111|
  PROCEDURE LERREG (TOPO : POINTER);

    VAR
          DADO  :  REGISTRO;
          OK    :  BOOLEAN;

      |22222222222222222222222222222|

  PROCEDURE SALTABR (VAR F : TEXT);
     |IGNORA BRANCOS|
   VAR
      FEITO          :  BOOLEAN;
   BEGIN
     FEITO           : = EOF (F);

     WHILE NOT FEITO DO
       BEGIN
         FEITO := (F ^> ESPACO) OR EOLN (F);

         IF NOT FEITO THEN
           BEGIN
             GET (F);
             FEITO := EOF (F);
           END;
       END;

   END; SALTABR
     |22222222222222222222222222222|

  PROCEDURE LERCARD (VAR F : TEXT;
                     VAR N : CARDINAL;
                     VAR OK : BOOLEAN);
     |OBTEM UM NUMERO POSITIVO|
   TYPE
        NUMERO = 0 .. 9;
   VAR
        DIGITOS :  SET OF CHAR;
        FEITO   :  BOOLEAN

        |33333333333333333333333|

   FUNCTION VALOR (DIGITO : CHAR) : NUMERO;
     |CONVERTE CHAR PARA VALOR NUMERICO|
     BEGIN
       VALOR := ORD(DIGITO) - ORD('0');

     END; |VALOR|

       |33333333333333333333333|

   FUNCTION LEGAL (N   : CARDINAL;
                   DIGITO : CHAR)        : BOOLEAN;
      |PROXIMIDADE A OVERFLOW DE CARDINAL|
     BEGIN
      IF N = MAXINT DIV 10
        THEN
          LEGAL := VALOR(DIGITO) < = MAXINT MOD 10
        ELSE
          LEGAL := N < MAXINT DIV 10

      END; |LEGAL|
        |3333333333333322222222222222|

   BEGIN     |LERCARD|

     DIGITOS := ['0' .. '9'];
     OK      := NOT EOF (F);

     IF OK THEN |E UM DIGITO?|
       OK := F^ IN DIGITOS;

     N := 0;
     FEITO := NOT OK;

     WHILE OK AND NOT FEITO DO
       BEGIN |N PARA A BASE 10 ||
         N := 10 * N + VALOR (F^);
         GET (F);
         FEITO := EOF (F);

         IF NOT FEITO THEN
           FEITO := NOT (F^ IN DIGITOS);

         IF NOT FEITO THEN
           OK := LEGAL (N, F^)
       END;

     IF NOT EOF (F) THEN
       READLN (F)

   END; |LERCARD|
     |22222222222222222222222222222|

  PROCEDURE LERLINHA (VAR S : STRING);
     |LE UMA LINHA DO TECLADO|
   VAR
        INDICE : 0 .. MAXSTRING
        SIMBOLO : CHAR
   BEGIN
     SALTABR (INPUT);
     INDICE := 0;

     WHILE NOT EOLN AND (INDICE < MAXSTRING) DO
       BEGIN |INSERE 1 CHAR NO STRING|
         INDICE := SUCC (INDICE);
         READ (SIMBOLO);
         S[INDICE] := SIMBOLO;
       END;

     IF INDICE < MAXSTRING
       THEN |COMPLETA COM NULOS|
         FOR INDICE := INDICE + 1 TO MAXSTRING DO
           S[INDICE] := CHR (0)
       ELSE
         IF NOT EOLN THEN |EXCESSO DE CHARS|
           WRITELN ('ATENCAO - TRUNCADO');

     READLN;

   END; |LERLINHA|
     |22222222222222222222222222222|

  PROCEDURE INSERE (LISTA : POINTER;
                    DADO : REGISTRO);
   VAR
        PPROX,
        LIGA : POINTER;
        ALFA : STRING;
   BEGIN |COLOCA DADO NO TOPO DA LISTA|
     LISTA ^. ITEM := DADO;
     PPROX := LISTA;
     ALFA := PPROX ^. PROXIMO ^. ITEM . NOME;

     WHILE DADO . NOME < ALFA DO
       BEGIN |PESQUISA NA LISTA|
         PPROX := PPROX ^. PROXIMO;
         ALFA := PPROX ^. PROXIMO ^. ITEM . NOME
       END;
         |SEM VERIFICACAO DE DUPLICIDADE|
     NEW (LIGA); |CRIA UM NO|
     LIGA ^. ITEM := DADO; |INSERE DADOS|
     LIGA ^. PROXIMO := PPROX ^. PROXIMO; |E LIGA|
     PPROX ^. PROXIMO := LIGA |A LISTA|

   END; |INSERE|
     |222222222222222221111111111111111|

  BEGIN |LERREG|

    WRITE ('NOME? => ');
    LERLINHA (DADO . NOME);
         |PARA NUMA ENTRADA NULA|
    WHILE DADO . NOME [1] <> CHR (0) DO
      BEGIN
        REPEAT
          REPEAT
            WRITE ('DEBITO? =>' : 15);

            IF EOLN (INPUT) THEN
              READLN (INPUT);

            SALTABR (INPUT)

          UNTIL NOT EOLN (INPUT);

          LERCARD (INPUT, DADO . DEBITO, OK);

          IF NOT OK THEN |SEM DIGITOS|
            WRITELN ('*** ERRO -  : 20,
              'REENTRE ***');

        UNTIL OK

        INSERE (TOPO, DADO);
        WRITELN ('(RETURN SE ACABOU)' : 40);
        WRITE ('NOME?');
        READLN (DADO . NOME);
      END

  END; |LERREG|
  |1111111111111111111111111111111111111|

  PROCEDURE DISPLAY (LISTA : POINTER);

    VAR
            MOSTRA : POINTER;
    BEGIN |IMPRIME O TOPO DA LISTA|
      MOSTRA := LISTA ^. NEXT;
      PAGE (OUTPUT);
      WRITELN ('LISTA ORDENADA :');
      WRITELN;
            |EXISTEM MAIS ITENS A MOSTRAR?|
      WHILE NOT (MOSTRA = LISTA) DO
        BEGIN
          WITH MOSTRA ^. ITEM DO
            WRITELN (DEBITO : 8,NOME : 30);

          WRITELN;
          MOSTRA := MOSTRA ^. PROXIMO
        END

    END; |DISPLAY|
    |1111111111111111111111111111111111111|

  BEGIN |LISTACIRC - PROGRAMA PRINCIPAL|

    NEW (LISTA); |CRIA NO VAZIO NO TOPO|
    LISTA ^. PROXIMO := LISTA; |QUE SE APONTA|
    PAGE (OUTPUT);
    WRITELN ('ENTRE COM OS DADOS :          SOBRENOME',
             'ANTES, DEPOIS COM O DEBITO :');
    WRITELN;
    LERREG (LISTA); |COLOCADOS NA LISTA|
    DISPLAY (LISTA); |MOSTRA EM ORDEM|
    WRITELN ('——— ENCERRADO ———' : 40)
  END.
```

EPSON PX-8



Alimentado por bateria, o Epson PX-8 possui RAM de 64 Kbytes, CP/M e diversos softwares gravados em chips de ROM, que o convertem num dos mais competitivos micros portáteis do mercado internacional.

Acondicionado num estojo do tamanho de uma lista telefônica, o PX-8 não se parece com um computador, graças a suas reduzidas dimensões e seu pequeno peso (cerca de 2,3 kg). No entanto, deslizando-se a tampa corrediça do estojo, surgem um teclado completo e um visor de cristal líquido (LCD, Liquid Crystal Display), dobrado, que permite onze inclinações diferentes, conforme o ângulo mais adequado ao operador. Sob o visor, encontra-se o gravador de microfita cassete.

O teclado, tipo máquina de escrever, apresenta 72 teclas, codificadas em cores para indicar suas finalidades. As alfanuméricas, em marrom-escuro, estão distribuídas no formato padrão QWERTY, nas versões americana e inglesa. Nesta, o caractere $ substitui o # (hash mark, "cancelamento") encontrado no modelo americano. Todos os caracteres de outras línguas podem ser produzidos pelo teclado, com o simples acionamento de algumas chaves.

Quatro teclas laranja controlam os movimentos do cursor. Há também as teclas [Insert], [Delete] e [Home], e três outras que acionam funções do sistema ([Escape], [Pausé] e [Help]), além de cinco para funções programáveis pelo usuário.

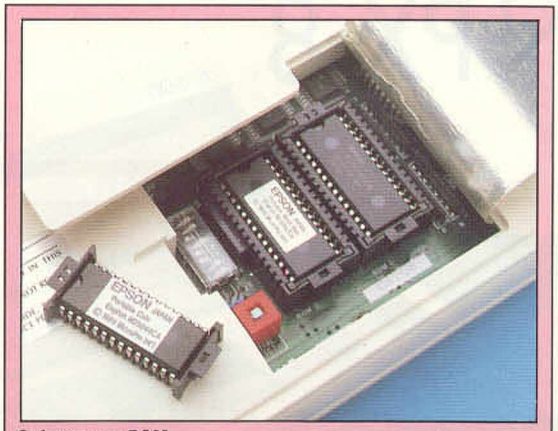
A operação do PX-8 é fácil com o aparelho colocado no colo, devido a seu design compacto. Torna-se menos funcional, no entanto, quando instalado sobre uma mesa, pois as teclas requerem pressão direta de cima para baixo. Duas pernas retráteis acopladas proporcionam certa inclinação à unidade, mas não resolvem satisfatoriamente o problema. As teclas [Caps Lock], [Number] e [Insert] servem para mudar os modos de tela do PX-8 — três pequenas luzes vermelhas no painel indicam o modo em uso.

A documentação, bastante abrangente, consiste em dois volumosos manuais: um, destinado ao usuário, trata da instalação do aparelho, uso do hardware, do software e das operações CP/M. O outro é um guia de referência para programação em BASIC.

O manual do usuário inclui também mapas da memória, listas completas dos caracteres disponíveis e respectivos códigos, além de um programa em ASSEMBLY bastante longo, para gravar e carregar gráficos de tela em disquete. O guia de referência começa explicando como instalar e utilizar o BASIC (fornecido, assim como o pacote

Boa opção
O micro portátil PX-8 é fabricado pela Epson, famosa por suas impressoras matriciais. Esse equipamento tem 64 Kbytes de RAM, visor de cristal líquido, CP/M e vários pacotes de software residentes em chips de ROM.

Grande LCD
O visor de cristal líquido, de 80 x 8 caracteres, utiliza a bateria do micro e oferece resolução gráfica de 480 x 64 pixels.



Software em ROM
Retirando-se um pequeno painel da parte inferior do PX-8, aparecem os conectores onde se ligam os softwares gravados em EPROM. Aí está instalado o WordStar, assim como o sistema operacional CP/M. Para passar do WordStar ao Calc, basta trocar as EPROMs.

de software, num chip ROM). O outro volume discorre sobre a natureza da programação, num exame dos vários modos de tela do PX-8, além de explicar em detalhe todos os comandos BASIC disponíveis.

Por dentro do Epson

O PX-8 utiliza CPU de CMOS (Complementary Metal Oxide Semiconductors, "semicondutores complementares de óxido metálico") compatível com o Z80. Os chips CMOS exigem muito menos energia do que os comuns. Acrescido ao baixo consumo do visor, o emprego destes chips permite o funcionamento da unidade com fonte de alimentação recarregável.

Acompanham o micro duas baterias, sendo uma de reserva. Garantidas por até quatro anos, devem ser carregadas antes de o computador entrar em uso; assim, é preciso esperar oito horas a partir da instalação do equipamento até o início da operação. A carga é suficiente para quinze horas de operação contínua.

Logo que o PX-8 estiver pronto para funcionar, seu sistema operacional deve ser inicializado. Os passos necessários para isso incluem entrar com dia, data, hora, e cuidar de algumas tarefas de manutenção.

Uma delas é a formatação de um disco simulado em RAM. O PX-8 reserva uma porção da RAM — selecionável pelo usuário entre 9 e 24 Kbytes —, tratando-a como um disco de armazenamento (RAMDISK). O sistema operacional utiliza essa área da memória como se fosse um disco externo. Antes de usá-lo, formata-se o RAMDISK, especificando-se a quantidade de RAM que se quer utilizar. A Epson também oferece uma unidade de disco adicional em RAM, contendo 120 Kbytes extras.

O PX-8 possui na memória o sistema operacional CP/M, residente em ROM, e exibe no LCD, sob forma de menu, uma lista de utilitários em CP/M e um diretório do software residente em ROM.

Softwares

O software, em cassete, disquete ou ROM, é gravado em chips EPROM que se encaixam num conector sob o micro. Os softwares que acompanham o PX-8 — WordStar (processador de texto), Calc (planilha) e Scheduler (banco de dados), em versões portáteis — são fornecidos nesse formato de módulo, assim como o interpretador BASIC. Para selecionar determinado aplicativo, utilizam-se as teclas do cursor para indicar a escolha desejada e aperta-se [Return]. Carrega-se então o programa selecionado da ROM para a RAM.

O visor de cristal líquido, de 80 x 8 caracteres, apresenta resolução gráfica de 480 x 64 pixels. A maior desvantagem desse tipo de tela está na lentidão. Embora os caracteres apareçam com rapidez satisfatória, à medida que são digitados, qualquer tipo de correção — sobretudo quando envolve palavras ou sentenças completas — leva um tempo excessivo para ser realizado.

O software fornecido com o PX-8 é de fácil compreensão. A Epson oferece também um programa de telecomunicações, para uso com modem, e um programa destinado a transferir arquivos do PX-8 para computadores maiores, como o QX10, da Epson. Funcionando com CP/M, o PX-8 aceita outros softwares desenvolvidos nesse sistema.

O BASIC do PX-8 é uma versão Microsoft melhorada pela Epson. Inclui o comando AUTO para numerar e renumerar as linhas dos programas, um editor de tela completo, bem como comandos gráficos e sonoros, para comunicações através da interface RS232C incorporada, além de comandos que permitem usar o gravador microcassete como unidade de disco.

Em resumo, o Epson PX-8 é um microcomputador ideal para jornalistas, homens de negócios e todos que precisam de um equipamento pequeno mas eficiente, de fácil transporte em viagens e utilizável em qualquer local, mesmo os que não possuem rede elétrica.

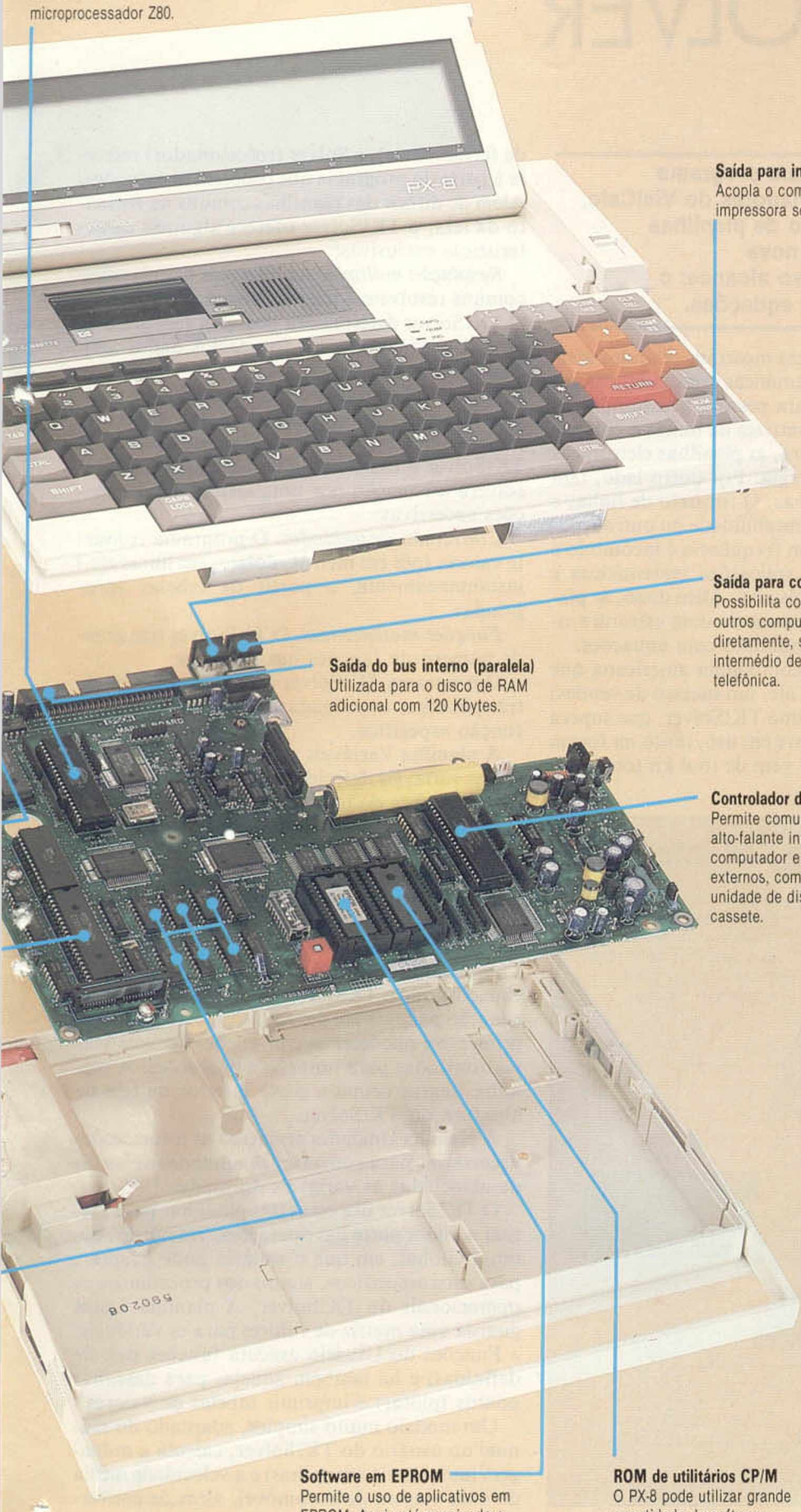
Saída para alto-falante
Amplifica o som quando em conexão com um alto-falante externo.

Saída A/D
Liga dispositivos externos como voltímetros, medidores de temperatura ou balanças.

Saída para leitora de código de barras
Permite o uso do PX-8 para controle de preços e de estoques.

Processador auxiliar 7508
Converte em digitais os sinais analógicos recebidos pelo conector A/D.

RAM
O PX-8 contém os 64 Kbytes de RAM exigidos pelo CP/M. A bateria de reserva assegura a manutenção do conteúdo da memória quando se desliga o aparelho.



CPU
É a versão CMOS do familiar microprocessador Z80.

Saída para impressora
Acopla o computador a uma impressora serial.

Saída para comunicações
Possibilita comunicação com outros computadores, seja diretamente, seja por intermédio de modem e rede telefônica.

Saída do bus interno (paralela)
Utilizada para o disco de RAM adicional com 120 Kbytes.

Controlador de periféricos
Permite comunicação com o alto-falante interno do computador e com periféricos externos, como impressora, unidade de disco ou gravador cassete.

Software em EPROM
Permite o uso de aplicativos em EPROM. Aqui está encaixado o WordStar, que pode ser substituído por outro módulo, como o Calc.

ROM de utilitários CP/M
O PX-8 pode utilizar grande quantidade de software disponível em CP/M.

EPSON PX-8

MICROPROCESSADOR
CPU, de CMOS, compatível com o Z80.

CLOCK
2,4 MHz.

MEMÓRIA
64 Kbytes de RAM, 32 Kbytes de ROM, mais 6 Kbytes de memória de vídeo.

SISTEMA OPERACIONAL
CP/M.

VÍDEO
Visor de cristal líquido. Modo texto, 80 x 8 caracteres; modo gráfico, 480 x 64 pixels.

INTERFACES
Saída para impressora serial RS232C; saída para leitura de código de barra; entrada analógica.

LINGUAGENS
BASIC em versão Microsoft melhorada, operando em CP/M.

TECLADO
Estilo máquina de escrever QWERTY, 72 teclas, incluindo quatro para controle do cursor e cinco para funções programáveis.

DOCUMENTAÇÃO
Dois grandes volumes encadernados em espiral: um manual de operações e um guia de referência de BASIC.

TK! SOLVER

Examinamos aqui um programa elaborado pelos criadores do VisiCalc, que leva o conceito de planilhas eletrônicas a uma nova direção e a um novo alcance: o processamento de equações.

As planilhas eletrônicas mostram-se muito úteis em várias tarefas matemáticas executadas em microcomputadores. Para pessoas acostumadas a manipular grandes matrizes de linhas e colunas com lápis e calculadora, as planilhas eletrônicas poupam tempo e energia. Por outro lado, têm limitações significativas. O formato de linhas e colunas, ideal para contabilidade ou outros modelos financeiros, com freqüência é incômodo e às vezes inútil para aplicações matemáticas e científicas de nível mais alto. Além disso, as planilhas eletrônicas apresentam uma estrutura rígida demais para o trabalho com equações.

A Software Arts (companhia americana que criou a planilha VisiCalc, um sucesso de vendas) desenvolveu o programa TK!Solver, que supera as planilhas atualmente em uso, tanto na forma como na função. TK vem de tool kit (conjunto de ferramentas), e Solver (solucionador) refere-se à parte do programa que processa as equações. Além de diferir das planilhas comuns no formato da tela, o TK!Solver oferece algumas características exclusivas:

Resolução múltipla. As fórmulas das planilhas comuns resolvem equações de uma só variável. O TK!Solver determinará qualquer incógnita de uma equação, se forem fornecidos dados suficientes para isso.

Iteração. Quando um valor necessário à resolução de uma equação falta ou é desconhecido, pode-se entrar com um valor estimativo. O TK!Solver o usará como ponto de partida e resolverá a equação por uma série de aproximações sucessivas.

Conversão de unidades. O programa converte valores (pés em metros, dólares em libras etc.) instantaneamente, a partir de tabelas apropriadas.

Funções matemáticas. O TK!Solver tem grande número de funções incorporadas.

O programa TK!Solver funciona por meio de três planilhas interligadas, cada qual com uma função específica.

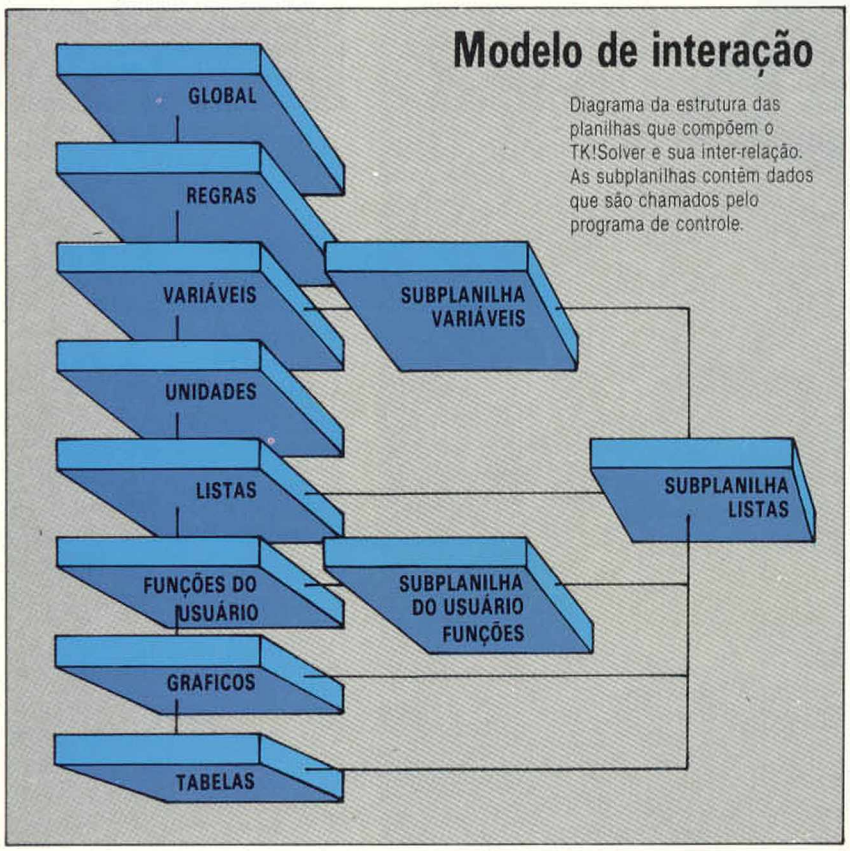
A planilha Variáveis contém os nomes de todas as variáveis definidas; colunas para os valores de entrada do usuário e para os de saída do programa; e espaço para indicar unidades associadas e para comentários sobre cada variável. Essa planilha aparece na parte superior da tela inicial do programa. Cada variável também é descrita em pormenores numa subplanilha Variáveis separada.

A planilha Regras é usada para introduzir as equações que o TK!Solver deve solucionar. Uma equação pode ter até duzentos caracteres de comprimento e obedecer às convenções matemáticas padronizadas para notação e operações. A planilha Regras ocupa a parte inferior da tela de abertura do TK!Solver.

A planilha Unidades armazena as informações necessárias para converter as unidades de medida associadas às variáveis do modelo.

O TK!Solver usa essas três planilhas para efetuar a maior parte das operações. Há outras, como a Global, em que o usuário pode adaptar, para usos específicos, alguns dos procedimentos operacionais do TK!Solver. A planilha Listas guarda uma matriz de valores para as variáveis; a Funções do Usuário executa funções por ele definidas; e há também aquelas para desenhar pontos (plotar) e imprimir tabelas de valores.

Um modelo muito simples, adaptado do manual do usuário do TK!Solver, calcula a milhagem (autonomia em milhas) e a velocidade média de uma viagem de automóvel, além de conver-



ter milhas em metros e seus múltiplos. Na tela de abertura do TK!Solver encontramos o cursor na planilha Regras, na parte inferior do vídeo. Começamos definindo as variáveis em equações adequadas teclando

Distância/Tempo = Velocidade

e pressionando [Return]. O TK!Solver transfere os nomes das variáveis das equações para a planilha Variáveis, na parte superior; examina a equação; e imprime as variáveis na coluna Nomes da planilha Variáveis, na mesma ordem em que aparecem na equação.

Um asterisco na coluna Status (estado) significa que a equação não foi satisfeita, pois não se introduziram valores em Variáveis. Entramos então com a segunda equação:

Distância/Combustível = Milhagem

Fornecida esta fórmula, todas as cinco variáveis definidas serão listadas na coluna Nomes da planilha Variáveis, conforme a *Tela 1*.

Tela 1: Equações e variáveis

Pressionando a tecla [;], movemos o cursor da planilha Regras para a Variáveis, onde podem ser introduzidos novos valores. O cursor surge na coluna de entrada ao lado da primeira variável, Distância. Digitamos então os valores no local adequado, conforme a tabela *à direita,* e acionamos [Return] ou a seta que aponta para baixo. Os valores Velocidade e Milhagem ficam em branco, pois cabe ao TK!Solver encontrá-los. Seus valores calculados serão apresentados na coluna Output (saída).

Para determinar a velocidade e a milhagem, pressionamos o ponto de exclamação [!], que o TK!Solver entende como tecla de "ação". Aparecerá a expressão Direct Solver acima da planilha Variáveis, indicando que o programa já recebeu os dados necessários para obter uma solução direta. Os valores Velocidade e Milhagem serão apresentados como saída. Apagamos então o que introduzimos anteriormente e obtemos um valor para Distância, fornecendo ao TK!Solver novos valores de velocidade e tempo, ou de milhagem e combustível.

Os valores introduzidos até agora não têm unidades associadas. Não podemos simplesmente digitar milhas, ou galões, na coluna Unidades da planilha Variáveis, pois as unidades não são utilizadas antes de definidas. Movimentamos o cursor para a planilha Regras pressionando [;] e depois digitando =U (de "unidade"). O TK!Solver substitui a planilha Regras, na janela inferior, pela Unidades. Esta tem quatro colunas: From (de); To (para); Multiply by (multiplicar por); e Add Offset (somar a constante).

Com o cursor sob FROM, podem-se introduzir as unidades que o TK!Solver deve reconhecer e os fatores de conversão, conforme a *Tela 2*.

Tela 2: Conversão de unidades

Digita-se =R para que volte à tela a planilha Regras, e a seguir [;], que move o cursor para a planilha Variáveis. Então, introduzem-se na coluna Unidades os nomes que definimos: m (milhas) para Distância; h para Tempo; m/h para Velocidade; g (galões) para Combustível; e m/g para Milhagem. Podemos agora apagar todos os valores atuais, substituindo-os por outros como: 1.247 para Distância, 22,5 para Tempo e 43,9 para Milhagem. Pressionamos [!] e a tela mostrará os valores no sistema métrico. *(Tela 3.)*

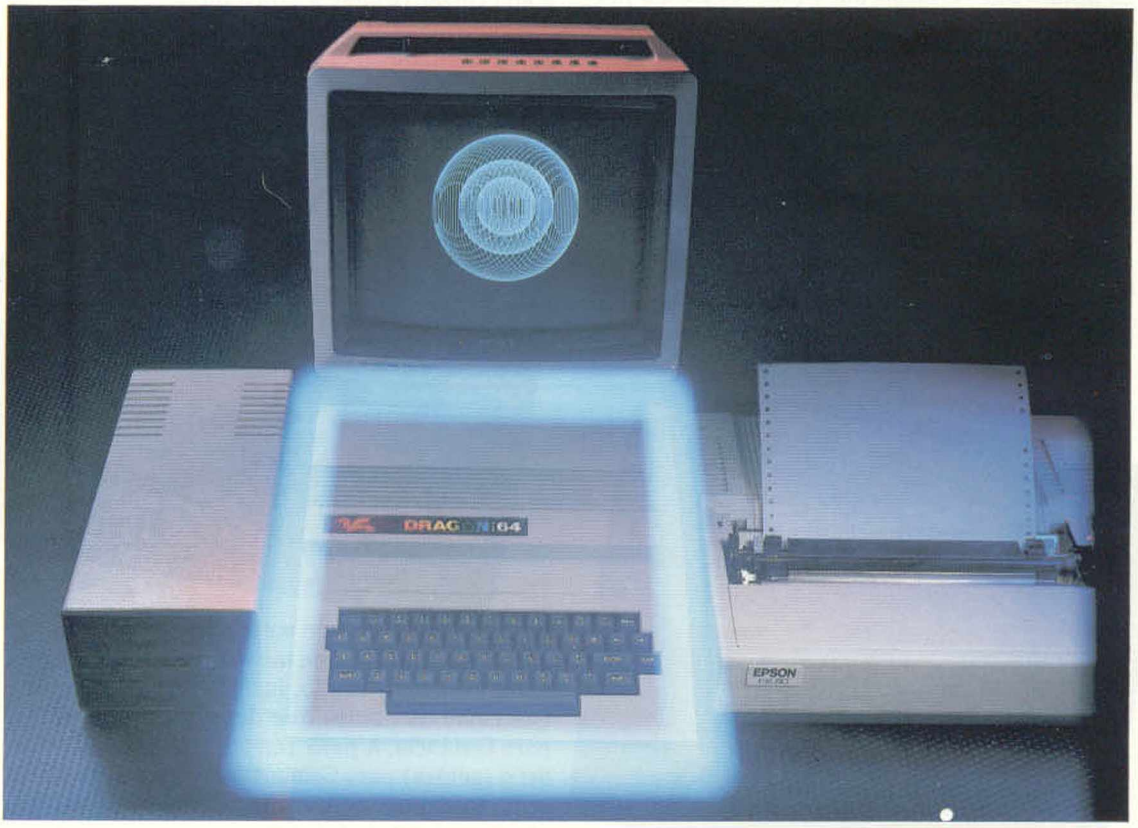
Tela 3: Conversão ao sistema métrico

Colocamos o cursor sobre o m na coluna de unidades, digitando km (quilômetros). Ao pressionarmos [Return], o TK!Solver converterá o valor 1.247 da Distância de milhas para quilômetros, passando esse valor para 2.006,423.

TABELA DE VALORES	
ENTRADA	NOME
500	Distância
8,5	Tempo
	Velocidade
14	Combustível
	Milhagem

OPERADOR DISCRETO

O gerente
Todo computador tem alguma forma de sistema operacional: um programa que gerencia o funcionamento da máquina e controla todos os dispositivos ligados ao sistema.



O sistema operacional trabalha "nos bastidores" de um computador: muitos usuários nem sabem que ele está presente na máquina. No entanto, é uma parte vital, constituindo o elo entre o hardware e o software.

As linguagens de alto nível permitem que o operador se isole das operações internas da CPU e contribuem para o maior intercâmbio de programas de uma família a outra. Desde que uma linguagem seja razoavelmente padronizada, as instruções devem valer para qualquer máquina que a aceite. O interpretador ou o compilador que processa o código-fonte de linguagem de alto nível também é um programa, devendo ser introduzido na memória principal antes de poder converter o código-fonte em instruções de código-objeto prontas para execução. O BASIC em ROM está presente na memória em caráter permanente, apto para utilização logo que se liga a máquina.

No entanto, para operar um computador, é preciso mais que um programa capacitando-o a converter o código-fonte em código de máquina. Deve haver outro programa que funcione em segundo plano, "nos bastidores", e se ocupe da organização interna. Consideremos, por exemplo, o problema de fazer aparecer no vídeo uma letra digitada no teclado. É preciso que haja, em algum ponto da memória, um programa que mande a CPU verificar constantemente o teclado para checar se qualquer tecla foi acionada. Se isso acontecer, o programa tem de identificar a tecla e, então, instruir os circuitos de vídeo para produzirem o padrão correto de pontos, na seqüência certa, para saída em vídeo. Tais atividades são chamadas "transparentes", isto é, invisíveis para o usuário.

Por exemplo, quando um comando é chamado para gravar um arquivo em fita cassete, o programador não se preocupa em saber como os dados se convertem numa forma adequada para armazenamento. Isso se realiza pelo sistema operacional — um programa gerenciador em processamento contínuo, encarregado de supervisionar todas as atividades do computador.

Surge uma leve fonte de confusão quando se utiliza um sistema de computação pequeno, baseado em ROM com BASIC residente, pois, nesse caso, o BASIC e o sistema operacional muitas vezes estão na mesma ROM. Em sua forma mais simples, uma ROM interna contém todo o soft-

ware básico necessário para operar o sistema, independentemente de programas aplicativos (jogos, processadores de texto etc.) carregados ou escritos pelo usuário.

Parte da ROM contém o interpretador, isto é, o código necessário para converter programas aplicativos escritos em BASIC para a linguagem de máquina (compreensível pelo microprocessador); outra parte contém o código necessário para introduzir e modificar programas escritos pelo usuário (o editor); e uma terceira parte contém o monitor, o software de organização interna necessário para varrer constantemente o teclado, apresentar na tela caracteres e desenhos, aceitar dados de fita cassete, alocá-los nos pontos corretos da memória e assim por diante.

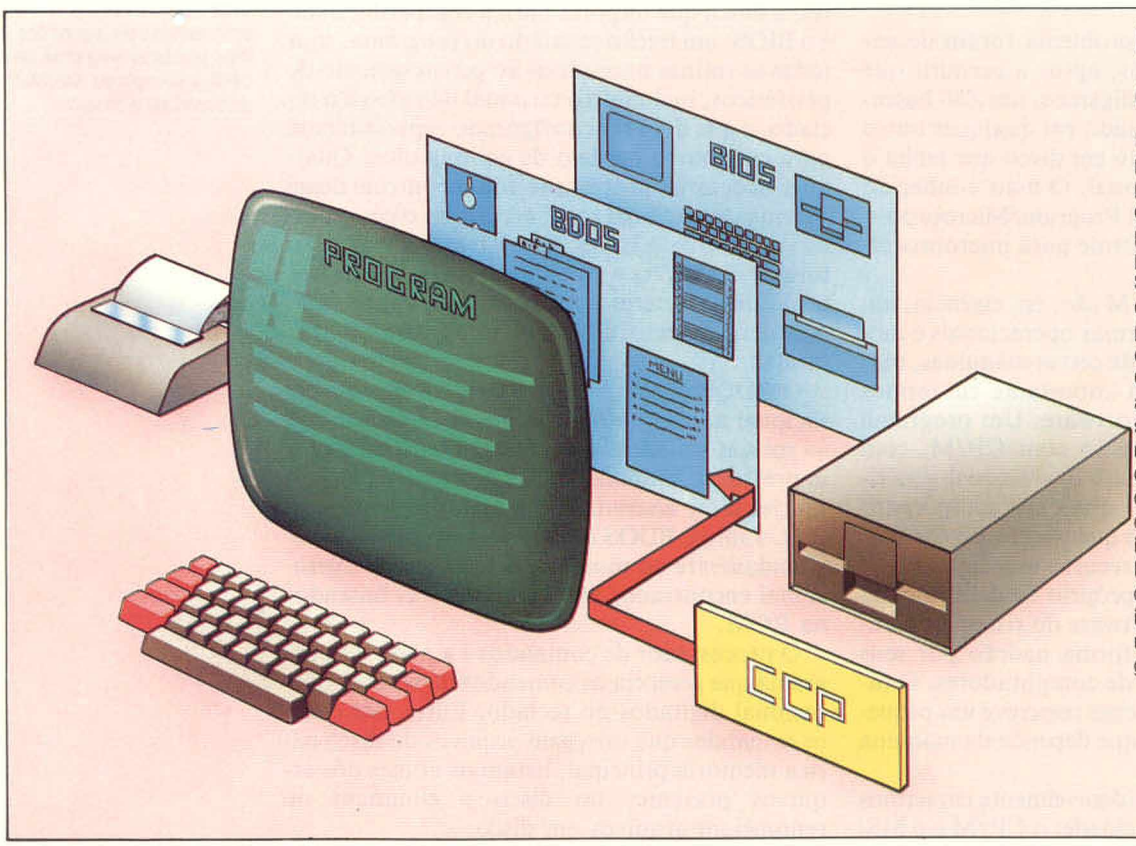
O termo "monitor" não se refere a qualquer componente de um sistema de vídeo: é quase um sinônimo de "sistema operacional". Em sua versão mais simples, o monitor pouco mais pode fazer do que aceitar instruções em linguagem de máquina, colocá-las no ponto correto da memória e supervisionar sua execução.

No outro extremo estão os sofisticados computadores profissionais baseados em discos, que dispõem de poderosos sistemas operacionais.

Um sistema baseado exclusivamente em discos flexíveis em geral tem poucas instruções armazenadas na ROM em caráter permanente, à exceção de um autocarregador e de algumas rotinas de organização interna. Quando o computador é ligado, o autocarregador contém instruções em linguagem de máquina suficientes apenas para instruir a CPU no sentido de acessar uma unidade de discos e carregar na memória principal o sistema operacional. Este, carregado na RAM, constitui o DOS (Disk Operating System, "sistema operacional de disco"), de recursos bem maiores que os sistemas baseados na ROM. O DOS amplia as funções normais de organização interna, acrescentando comandos que atuam diretamente sobre os arquivos armazenados em discos. Os comandos de um DOS podem, por exemplo, listar os nomes dos arquivos contidos no disco, eliminá-los ou renomeá-los e copiar arquivos em disco na memória principal ou em outros discos.

Mais convencionais, os sistemas de operação baseados em ROM quase nunca dispõem desses comandos de processamento de arquivo. Em certos casos, podem ter simplesmente um comando para carregar arquivos nomeados a partir de fitas ou armazenar em fita um arquivo sob determinado nome. Sistemas operacionais sofisticados conhecem o endereço exato em que se armazena qualquer arquivo, seja o meio um disco ou uma fita. Sistemas menos avançados podem ser capazes apenas de fazer uma pesquisa sistemática em todos os arquivos existentes, até encontrar o nome desejado e então carregá-lo; ou gravar um arquivo sob determinado nome na fita cassete, no ponto em que esta estiver quando o comando for chamado.

Mas não existem apenas os sistemas convencionais, baseados na ROM, e os baseados em discos flexíveis; há uma vasta gama de computadores intermediários. Um bom exemplo é o Apple II, apto a manipular arquivos em fitas ou discos, muitas vezes a partir da utilização dos mesmos comandos. O sistema operacional



Controles operacionais

No sistema operacional CP/M, o programa do usuário é processado por meio do BDOS, coletânea de rotinas que proporciona recursos e funções padronizados para o programa. Sendo o BDOS o mesmo em qualquer máquina, um programa que o utilize funciona em qualquer sistema CP/M. Quando o BDOS precisa efetuar alguma tarefa, chama o BIOS, sistema adequado para cada máquina e que representa o elo de software entre o BDOS e os componentes físicos do computador. O CP/M também inclui o CCP (Command Control Program, "programa controlador de comando"), que permite a introdução de comandos para rodar programas e, de modo geral, controlar o sistema.

é residente em ROM, mas está fisicamente separado; pode-se imaginá-lo como um sistema operacional que foi carregado na memória a partir de dispositivos externos de armazenamento. Suas funções incluem comandos de gerenciamento de arquivos bem mais avançados que os encontrados no Spectrum e em outros computadores baseados somente em ROM.

Compatibilidade

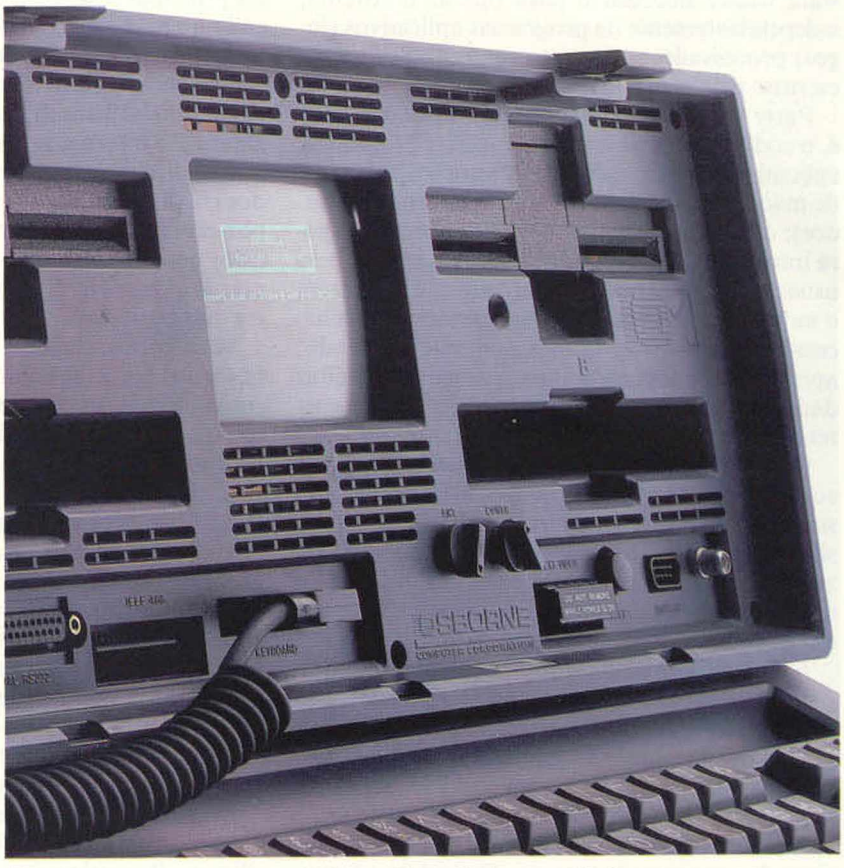
A capacidade de usar o mesmo software em mais de uma família de computadores é conhecida como "compatibilidade". Nela, em geral, existem dois aspectos. O primeiro consiste em diferentes processadores requererem diferentes conjuntos de instruções para efetuar operações equivalentes. Assim, as instruções em linguagem de máquina necessárias para, por exemplo, somar o conteúdo de dois endereços de memória teriam uma forma se fossem escritas para o processador 6502, usado no Apple, e outra inteiramente diversa para o Z80, o chip do CP 500. O problema da conversão de linguagens de alto nível em linguagem de máquina é, contudo, de responsabilidade do interpretador ou compilador utilizado: CPUs diversas requerem interpretadores e compiladores diferentes.

O segundo aspecto a afetar a compatibilidade do software está no fato de que o uso da mesma CPU — no Unitron AP II e no Microdigital TK 2000, por exemplo — não impede a manifestação de outras dificuldades: uso de diferentes endereços para a memória de vídeo; necessidade de diferentes códigos para movimentar o cursor; existência de diferentes recursos de entrada e saída etc.

Para fazer frente ao problema foram desenvolvidos DOSs genéricos, aptos a permitir que o software escrito para, digamos, um Z80 baseado em disco seja processado em qualquer outro computador Z80 baseado em disco que tenha o mesmo sistema operacional. O mais conhecido DOS é o CP/M (Control Program/Microprocessors, "programa de controle para microprocessadores").

Os DOSs do tipo CP/M são, em essência, um desenvolvimento de sistemas operacionais e monitores mais específicos de certas máquinas, mas representam um avanço importante em termos de compatibilidade de software. Um programa escrito para processamento com CP/M, com MS-DOS ou outro sistema operacional genérico, será processado por qualquer computador com esses sistemas, desde que o software não tente fazer uso de nenhum recurso especial (efeitos sonoros, por exemplo) próprio de determinada máquina. O próprio software do sistema operacional é fornecido em forma padrão por seus criadores ao fabricante de computadores. O fabricante de hardware apenas reescreve um pequeno trecho do programa que depende da máquina a ser utilizada.

Os DOSs variam consideravelmente em termos de complexidade e capacidade; o CP/M e o MS-DOS estão entre os mais simples. Compõem-se, no fundamental, do processador de comandos, do BDOS (Basic Disk Operating System, "DOS básico") e do BIOS (Basic Input/Output System, "sistema básico de entrada e saída"). Dessas partes, a única que importa para a compatibilidade é o BIOS: um trecho separado do programa, com todas as rotinas necessárias ao gerenciamento de periféricos, incluindo o terminal de vídeo e o teclado, e que deve ser configurado especialmente para cada novo modelo de computador. Qualquer programa processado sob o controle desse sistema operacional interfaceia com o computador por meio do BIOS. Este vai comandar a leitura de caracteres a partir do teclado, a apresentação de caracteres no vídeo ou na impressora, o endereçamento de discos e inúmeras outras tarefas.



A chave do sucesso
O Osborne-1 deve boa parte de seu êxito de vendas ao fato de ter sido o primeiro microcomputador portátil a usar o sistema operacional CP/M. Cada máquina vem acompanhada por alguns dos mais populares programas para CP/M, a exemplo do WordStar, do SuperCalc e do MBASIC.

O BDOS consiste nas partes do sistema operacional não específicas ao nível de dispositivo — rotinas genéricas de gerenciamento de vídeo, impressoras, unidades de disco etc. — e que não precisam ser alteradas para novas implementações. Juntos, BDOS e BIOS correspondem aproximadamente ao monitor ou ao sistema operacional encontrados em computadores baseados na ROM.

O processador de comandos é a parte do programa que gerencia os comandos do sistema operacional digitados no teclado. Entre eles estão os comandos que carregam arquivos do disco para a memória principal, listam os nomes dos arquivos presentes no disco e eliminam ou renomeiam arquivos em disco.

HISTÓRIA DO MICROPROCESSADOR

Dos milhões de microprocessadores usados em todo o mundo, a maioria é de modelos 8088, 6502 e Z80. Embora curta, a história dos microcomputadores mostra como se estabeleceu essa hegemonia.

Os computadores operados por um único microprocessador surgiram quase por acidente. Em 1972, a Intel, fabricante americana de chips, foi encarregada pela Datapoint de desenvolver um chip que substituísse o processador discreto (específico), formado por grande número de chips TTL (Transistor-Transistor Logic, "lógica transistor a transistor") e utilizado nos terminais de computadores.

O produto desenvolvido foi o 8008, um chip capaz de processar dados a 8 bits por vez, mas que apresentava sério inconveniente: trabalhava muito devagar. A Datapoint decidiu não lançá-lo, mas o potencial do 8008 como uma CPU para computadores de uso geral não passou despercebido. Surgia o computador portátil, de baixo custo.

As evidentes limitações do 8008 levaram a Intel a projetar um modelo mais aperfeiçoado. Um novo chip, o 8080, logo se tornou a força dominante no mercado.

O lançamento do 8080 praticamente coincidiu com o do 6800, microprocessador também de 8 bits desenvolvido pela Motorola, concorrente da Intel. As filosofias de estrutura dos dois produtos divergiam bastante, mas ambos eram potentes e capazes de servir de base a um projeto de microprocessador.

O desenvolvimento seguinte veio em 1974, quando Gary Kildall, então presidente da Digital Research, projetou para a Intel o CP/M, um sistema operacional em discos. Isso permitiu que os computadores baseados no 8080 fossem usados com as unidades para discos flexíveis, recém-introduzidas no mercado. Apesar disso, a Intel rejeitou o sistema operacional CP/M, alegando que o software existente era mais que suficiente para os computadores de grande porte da época.

No entanto, os computadores menores estavam se tornando cada vez mais populares, e neles o CP/M facilitava muito a manipulação de arquivos. Isso assegurou por muito tempo o domínio do mercado pelo 8080 e colocou o Motorola 6800 em relativa obscuridade. Houve diversas tentativas de produzir sistemas operacionais em disco equivalentes para o 6800, mas a atenção já estava toda voltada para o 8080.

A escolha vital
A maioria dos micros pessoais utiliza um microprocessador 6502 (por exemplo, o Micro Engenho, da linha Apple) ou um Z80 (caso do TK85, da linha Sinclair, do TK90X, compatível com o Spectrum, e de outros micros da série TK).

À medida que crescia o mercado de produtos baseados em microprocessadores, aumentou bastante a competição entre os fabricantes de chips para o desenvolvimento de novos modelos. Deparavam, porém, com a relutância dos compradores em aceitar algo novo, a menos que fossem oferecidas nítidas vantagens. Os investimentos em hardware e software também contribuíam para desestimular a adoção de processadores incompatíveis com a tecnologia existente.

Um lance de genialidade abriu espaço para um novo modelo de chip — o Z80. A Zilog nasceu de uma equipe de engenheiros que trabalhara no 8080 da Intel; eles perceberam que o conjunto de instruções binárias do 8080 poderia ser ampliado. Não haviam sido exploradas todas as combinações possíveis de 1 e 0 que esse chip poderia reconhecer como instruções. Incorporando combinações binárias "esquecidas" pela Intel, a Zilog criou um microprocessador que operava de modo idêntico ao 8080 quando recebia instruções típicas desse microprocessador, mas que tinha melhor desempenho. Dessa maneira, tornou-se possível usar o software desenvol-

A evolução dos chips

Este diagrama mostra como os microprocessadores se desenvolveram e também algumas das máquinas em que foram utilizados. Diversos chips pouco conhecidos aparecem nos micros menos populares. O Apple III talvez seja o único micro profissional que use o processador 6502. O Olivetti M20 é o único micro de uso geral equipado com o chip Z8000. Em ambos os casos, a escolha de um microprocessador incomum e a conseqüente falta de software inibiram o êxito da máquina. Paralelamente o IBM PC e outros computadores de enorme sucesso estendiam sua popularidade aos chips utilizados por eles.

Intel

4004 → **8008** → **8080** → **8085** → **8086** → **8088**

4004 e 8008 Projetado para substituir processadores discretos compostos por grande número de circuitos integrados TTL (Transistor-Transistor Logic, "lógica transistor a transistor"), o chip 4004 podia manipular apenas palavras de 4 bits. O 8008, desenvolvido logo em seguida, processava os dados de 8 em 8 bits. Reconhecendo seu potencial, amadores e engenheiros começaram a utilizar esse chip na construção "caseira" de microcomputadores.

·'71 ·'72 ·'73 ·'74 ·'75 ·'76 ·'77 ·'78 ·'79 ·'80 ·'81 ·'82 ·'83 ·'84

Motorola

6800 → **6809** → **68000** → **68008**

6800 Este microprocessador tinha recursos semelhantes aos do 8080, desenvolvido pela Intel mais ou menos na mesma época, mas seu projeto e modo de operação eram totalmente diversos.

6809 Aperfeiçoando seu 6800, a Motorola criou o 6809, talvez o mais versátil dos chips de 8 bits. Mas os concorrentes já dominavam o mercado, e esse novo chip foi utilizado em poucas máquinas.

Dragon 32

68000 A eficácia do chip de 16 bits da Motorola foi amplamente reconhecida. A Sinclair utilizou em seu modelo QL o chip 68008, versão reduzida do 68000. Mas seu êxito acabou limitado pela falta de software barato e pelo domínio do 8088.

QL

Lisa

MOS Technology

6502

6502 A tecnologia MOS criou seu próprio chip de 8 bits, que, embora incompatível com o 6800, em grande parte derivava dele. Seu baixo custo atraiu o interesse de amadores e projetistas. O resultado foi seu emprego na primeira geração do Apple e do Commodore PET, máquinas de enorme sucesso comercial. O chip permanece popular, utilizado no Unitron, no Micro Engenho, no TK 2000 e em outros computadores.

PET

Apple III

IBM PC

Zilog

Z80 → **Z8000**

Z80 Uma equipe de projetistas deixou a Intel para formar a Zilog, que produziu esta versão aperfeiçoada e plenamente compatível com o 8080.

TRS-80 Model 1

Z8000 O primeiro chip de 16 bits da Zilog não se saiu muito bem no mercado dos micros. A segunda tentativa da empresa foi o Z800, que promete total compatibilidade com o Z80 (portanto, também em relação ao 8080, da Intel).

Olivetti M20

8080 Com a criação do sistema CP/M, tornou-se o primeiro chip apto a ser empregado em micros tanto de uso pessoal quanto profissional. Mais tarde, a Intel criou o 8085, com funções adicionais e menos chips auxiliares. O modelo 8085 é encontrado numa versão CMOS de baixo consumo, presente em portáteis.

TRS-80 Model 100

8088 e 8086 Uma versão reduzida, o 8088 pode usar chips auxiliares mais antigos; seu emprego no IBM PC fez dele o mais popular chip de 16 bits. O 8086, de melhor desempenho e totalmente compatível com o 8088, é hoje utilizado na maioria das máquinas.

vido para o 8080 e contornar o problema da compatibilidade.

Além dessas inovações, introduziu-se importante vantagem comercial. O chip da Intel dependia de um chip gerador de sincronismo (clock) e de um chip de controle de sistema, enquanto o Z80 combinava num único chip toda a lógica necessária a um computador. Embora fosse relativamente caro, a capacidade de substituir vários outros chips tornava-o vantajoso para muitos fabricantes.

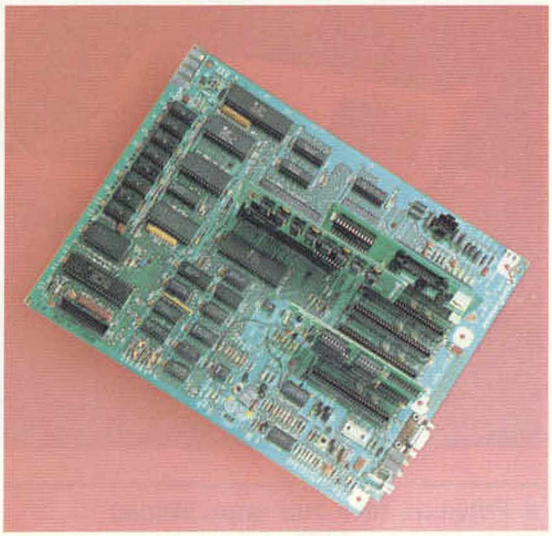
O chip 6800 não teve a mesma sorte em termos comerciais. A Motorola criou, afinal, um sofisticado microprocessador de 8 bits, o 6809, que representava um aperfeiçoamento do 6800. Quando o 6809 chegou ao mercado, porém, uma companhia rival, a MOS Technology, havia lançado o 6502, chip que constituía um aperfeiçoamento ainda maior do 6800. O modelo 6502 é o mais popular de uma série de microprocessadores conhecida como 6500, cujos integrantes usam o mesmo conjunto de instruções, mas diferem em potência e recursos.

O Z80 incorpora o conjunto de instruções do 8080 e pode substituí-lo num sistema de computação, embora exija, para tanto, modificações estruturais. O projeto do 6502, por sua vez, segue uma filosofia muito próxima à do Motorola 6800. Qualquer programador do 6800 estaria "em casa" com o conjunto de instruções do 6502. Todavia, este não é compatível com o 6800 quanto aos requisitos de hardware, software ou possibilidade de substituição chip por chip. Por tudo isso, era pouco provável que viesse a alcançar êxito imediato, exceto por outro golpe de sorte: foi adotado no computador Apple, muito bem-sucedido.

Quando a Apple surgiu, os micros portáteis eram dominados pelos projetos de bus baseados no S-100. (O bus diz respeito ao número de bits de um computador. Esse número pode ser dado pelo bus interno — por exemplo, 32 bits no 6800 — ou pelo bus externo — 16 bits, no mesmo 6800. O valor dos bus varia de acordo com o microprocessador; nesta matéria, não havendo indicação em contrário, adota-se o valor do bus interno.)

O S-100 é uma padronização da conexão entre elementos, na qual sinais eletrônicos se transmitem pela ligação de placas. Os micros baseados nesse sistema dispõem de inúmeras placas — a principal, que transporta a energia e os sinais, e várias outras, uma para cada função. Um sistema S-100 reduzido ao mínimo exigiria uma fonte de energia, a placa principal, outras para a CPU, a memória, o vídeo e, provavelmente, mais placas para a impressora e a unidade de disco. Nesse sistema, cada placa necessita de um conector, e o conjunto de conectores requer mais uma placa para a conexão geral. Tudo isso o torna bem mais caro que o sistema Apple, que contém, numa única placa, a CPU, o vídeo, a memória e as saídas para as demais expansões — conhecidas por slots.

A grande conquista dos criadores do Apple foi a planilha eletrônica VisiCalc. Esse aplicativo conquistou enorme popularidade entre os empresários: era mais fácil e mais rápido processar previsões financeiras com ele do que com calculadora, lápis e papel. O VisiCalc acarretou vendas em massa do computador Apple. Em conseqüência, o 6502 tornou-se um dos microprocessadores mais importantes.



Redução de chips
Chips mais sofisticados reduzem o número de componentes numa placa de circuitos. Quando a Apple aperfeiçoou o Apple II, a nova versão IIe tinha metade dos chips mais importantes.

Enquanto o chip 6502 consolidava seu domínio na área dos computadores de 8 bits, começavam a surgir no mercado os computadores de 16 e 32 bits. A Intel oferecia-lhes o 8086 e o 8068 (microprocessadores com bus interno de 16 bits e externo de 8 bits). A Motorola produzia o 68000 e a Zilog, o Z8000 (ambos com bus interno de 32 bits e externo de 16 bits). Nenhum dos modelos é compatível com seus predecessores de 8 bits. A Digital Research e a Microsoft criaram sistemas operacionais para os 8066/8088 (CP/M e MS-DOS, respectivamente) da Intel, mas a Zilog e a Motorola ainda não dispõem de sistemas operacionais adequados. Além disso, a adoção pela IBM do 8088 em seu PC consolidou a preferência pelo chip da Intel.

A luta entre os chips de 16 bits pelo domínio do mercado promete reeditar a história dos chips de 8 bits. O 8086 e sua versão reduzida, o 8088, tornaram-se padrões, da mesma forma que o Z80 e o 6502. Os principais motivos são o software de apoio fornecido pelos sistemas operacionais MS-DOS e CP/M-86 e sua escolha pelos micros de alta vendagem, sobretudo o IBM PC. O chip Zilog Z8000 tem sido utilizado apenas num micro de emprego geral, o Olivetti M20. Depois de enfrentar sérias dificuldades para fornecer software a esse micro, a Olivetti lançou uma placa conectável contendo um microprocessador 8086, possibilitando seu funcionamento com software para o MS-DOS e o CP/M-86.

Apesar do desenvolvimento acelerado das máquinas que utilizam microprocessadores de 16 bits, mais velozes e potentes, a maioria dos computadores se baseia nos chips de 8 bits, seja o Z80 ou o 6502.

POLIMAX

Equipamentos dedicados
O sistema Poliscriba 301 WP é o principal produto da Polymax na área do processamento de textos — uma especialidade da empresa, desde sua fundação.

Teclado inteligente
Além de operar como terminal para sistemas de grande porte, o Poly 201 DE — criado especialmente para aplicações financeiras — também pode ser usado como um micro convencional autônomo.

A Polymax teve sua implantação aprovada, em 1977, pela extinta Capre (Coordenadoria de Atividades de Processamento Eletrônico), precursora da SEI (Secretaria Especial de Informática).

A Polymax Sistemas Periféricos foi fundada, em agosto de 1977, pelos engenheiros eletrônicos Lawrence Huang e Marcos Lee. Pouco depois, entrou para a sociedade Sheun Ming Ling, que financiou e dirigiu a empresa até julho de 1984. Nesse ano, a companhia Invesplan assumiu o controle acionário da Polymax, efetuando alterações na estrutura da empresa.

Para aumentar o capital, foram colocadas no mercado novas ações da empresa, agora com a razão social modificada para Polymax Informática. As mudanças na composição do capital e da diretoria não afetaram a continuação das atividades da firma.

Até o início de 1979 a principal preocupação da empresa esteve voltada para sua capacitação tecnológica e formação de uma equipe técnica. Nessa fase desenvolveram-se dois protótipos, dos quais resultaram o Poly 101 HS, micro dedicado ao processamento de textos, e o Poly 101 SS, para processamento de dados. Ambos começaram a ser comercializados em janeiro de 1979.

Dois anos depois foram substituídos por duas novas versões: Poly 201 WP e Poly 201 DP. A produção do primeiro cedeu lugar à do Poly 301 WP. Muito semelhante ao anterior, o projeto desse novo equipamento destinava-se a elevar o índice de produtividade na edição, no tratamento e no gerenciamento de textos.

Por suas características, o Poly 301 WP adapta-se aos sistemas de rede local e à automação de escritório. Sua capacidade de armazenamento também é superior à dos modelos anteriores, permitindo o uso de discos flexíveis de 8 polegadas ou discos rígidos tipo Winchester.

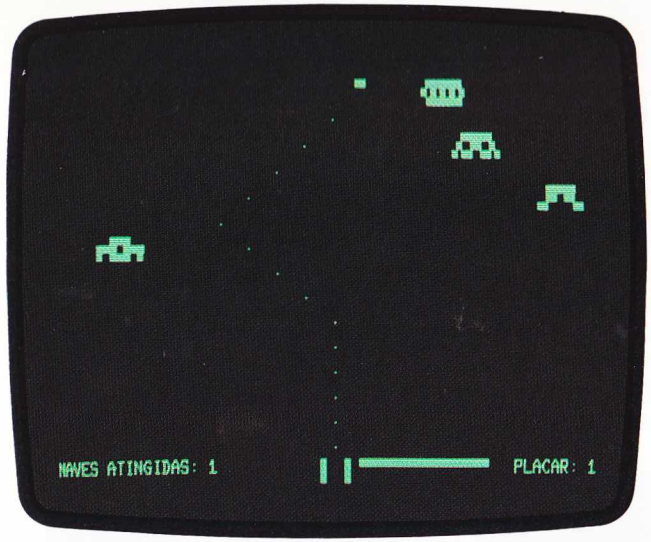
Durante o ano de 1982, a Polymax consolidou sua linha de equipamentos voltada ao processamento de textos e deu início a projetos de software para a área de teleprocessamento. Nessa linha se enquadra o lançamento do Sistelp (Sistema de Teleprocessamento Polymax), para operar com protocolos assíncronos e síncronos.

No mesmo ano, o mercado passou a contar com mais um microcomputador, o Maxxi, compatível com a linha Apple II Plus. O sistema multiusuário Poly 910 NET, lançado em abril de 1983, conectava até oito estações, configuradas por micros ou terminais inteligentes, permitindo o acesso a arquivos e troca de mensagens entre usuários ("correio eletrônico").

Nessa época, a Polymax também desenvolveu dois sistemas para atender o setor financeiro: o Poly 201 DE (Data Entry) e o Poly 105 MT. O primeiro é produto híbrido, empregado como terminal para entrada de dados do sistema central e como microcomputador. O segundo consiste num terminal inteligente.

Na linha dos periféricos, o produto mais importante é a impressora Polyprint 90, com velocidade de 90 cps. Ainda em 1983, chegava ao mercado o PEC 76, simulador de unidades de controle IBM 3276.

Já sob a coordenação da Invesplan, em abril de 1985, houve o lançamento do sistema Poly 920 NET, uma rede do tipo barramento (bus) que admite interconexão de até 255 estações configuradas por micros da Polymax ou terminais inteligentes.



Correção de trajetória
Se o lançamento do míssil não foi perfeito, você pode corrigir a trajetória com as teclas [Z] e [/].

Melhor desempenho
Começar derrubando as mais lentas e depois passar para as velozes é uma forma de desenvolver a técnica e estabelecer recordes.

```
    naves na tela. Elas se movimentam":
    PRINT TAB(5) "a uma velocidade que
    vai de 1 a 5. Os pontos ganhos";
200 PRINT TAB(5) "por cada nave abatida
    sao exatamente a sua velocidade,":
    PRINT TAB(5) "portanto procure
    acertar as naves mais rapidas."
210 PRINT:PRINT:PRINT TAB(17) "USE QQ
    TECLA PARA CONTINUAR";
220 I$="":I$=INKEY$:IF I$="" THEN 220
230 CLS
240 PRINT TAB(5) "Os misseis sao
    disparados atraves das teclas:":
    PRINT TAB(10) "Barra de
    Espacos  -  Disparo na Vertical"
250 PRINT TAB(10) "Seta <- ou Z -
    Disparo diagonal p/ Esquerda":
    PRINT TAB(10) "Seta -> ou / -
    Disparo diagonal p/ Direita"
260 PRINT TAB(5) "Voce controlara'
    o rumo do missil atraves dessas":
    PRINT TAB(5) "teclas tambem. Se
    voce for MUITO bom, podera' abater":
    PRINT TAB(5) "todos os inimigos de
    uma so' vez..."
264 PRINTTAB(17)"USE QQ TECLA PARA CONTINUAR"
265 I$="":I$=INKEY$:IF I$=""THEN265
269 CLS
270 PRINT TAB(5) "Quando terminar o jogo,
    serao mostrados o numero":PRINT
    TAB(5) "de pontos, o melhor placar e
    o numero de naves abatidas."
280 PRINT:PRINT TAB(19) "USE QQ TECLA
    PARA JOGAR";
290 I$="":I$=INKEY$:IF I$="" THEN 290
300 GOSUB 870
310 REM ***     DESENHOS DOS OBJETOS    ***
320 X$=CHR$(149)+" "+CHR$(170)
330 A$(0)=CHR$(166)+CHR$(183)+CHR$(183)+
    CHR$(183)+CHR$(132)
340 A$(1)=CHR$(174)+CHR$(187)+CHR$(187)+
    CHR$(187)+CHR$(132)
350 B$(0)=CHR$(152)+CHR$(179)+CHR$(143)+
    CHR$(179)+CHR$(164)
360 B$(1)=CHR$(168)+CHR$(163)+CHR$(143)+
    CHR$(147)+CHR$(148)
370 C$(0)=CHR$(176)+CHR$(143)+CHR$(131)+
    CHR$(143)+CHR$(176)
380 C$(1)=CHR$(140)+CHR$(179)+CHR$(131)+
    CHR$(179)+CHR$(140)
390 D$(0)=CHR$(156)+CHR$(174)+CHR$(179)+
    CHR$(157)+CHR$(172)
400 D$(1)=CHR$(141)+CHR$(174)+CHR$(179)+
    CHR$(157)+CHR$(142)
```

```
410 REM ***     INICIO DO JOGO          ***
420 CLS:D$=STRING$(15,131):GOSUB 740  :
    GOSUB 750  :GOSUB 760  :GOSUB 770  :
    PRINTà 993,D$;:F=0:S=990:F9=1
430 PRINTà 0,;
440 REM ***   VERIFICA   TECLADO       ***
450 IF PEEK(14656)=32 OR PEEK(15119)=4
    THEN F=1
460 IF PEEK(14656)=64 OR PEEK(14880)=128
    THEN F=2
470 IF PEEK(14912)=128 THEN F=3
480 REM ***    TRAJETORIA DO MISSIL    ***
490 PRINTà S,".";
500 IF F=1 THEN S=S-67 ELSE IF F=2 THEN
    S=S-61 ELSE IF F=3 THEN S=S-64
510 PRINTà 989,X$;
520 REM ***    MISSIL SAI DA TELA       ***
530 IF S<0 THEN FOR X=9000 TO 9008:
    L=USR(X):NEXT:CLS:F=0:S=990:D$=
    LEFT$(D$,LEN(D$)-1):IF LEN(D$)=0
    THEN 790  ELSE PRINTà 993,D$;
540 REM ***      ATUALIZA TELA          ***
550 PRINTà S,CHR$(131);:PRINTà 1009,
    "PLACAR:";SC;:IF S<>990 THEN PRINTà
    0,;:X=USR(9000)
560 PRINTà 960,"NAVES ATINGIDAS:";SH;
570 F9=1-F9
580 REM ***      ATINGIU  NAVE          ***
590 IF S>A1-3 AND S<A1+5 THEN PRINTà A1,
    "B U M!";:PRINTà 0,;:X=USR(40):
    PRINTà A1,"       ";:SC=SC+B1:SH=SH+1:
    GOSUB 740
600 IF S>A2-3 AND S<A2+5 THEN PRINTà A2,
    "B U M!";:PRINTà 0,;:X=USR(45):PRINTà
    A2,"       ";:SC=SC+B2:SH=SH+1:
    GOSUB 750
610 IF S>A3-3 AND S<A3+5 THEN PRINTà A3,
    "B U M!";:PRINTà 0,;:X=USR(50):PRINTà
    A3,"       ";:SC=SC+B3:SH=SH+1:
    GOSUB 760
620 IF S>A4-3 AND S<A4+5 THEN PRINTà A4,
    "B U M!";:PRINTà 0,;:X=USR(55):PRINTà
    A4,"       ";:SC=SC+B4:SH=SH+1:
    GOSUB 770
630 REM ***      MOVIMENTA NAVES        ***
640 PRINTà A1,"       ";:A1=A1+B1:PRINTà
    A1,A$(F9);
650 PRINTà A2,"       ";:A2=A2-B2:PRINTà
    A2,B$(F9);
660 PRINTà A3,"       ";:A3=A3+B3:PRINTà
    A3,C$(F9);
670 PRINTà A4,"       ";:A4=A4+B4:PRINTà
    A4,D$(F9);
680 IF A1>62 THEN PRINTà A1,"       ";:
    GOSUB 740
690 IF A2<128 THEN PRINTà A2,"       ";:
    GOSUB 750
700 IF A3>318 THEN PRINTà A3,"       ";:
    GOSUB 760
710 IF A4>446 THEN PRINTà A4,"       ";:
    GOSUB 770
720 GOTO 430
730 REM ***     VELOCIDADE DAS NAVES    ***
740 A1=0:B1=RND(5):RETURN
750 A2=191:B2=RND(5):RETURN
760 A3=256:B3=RND(5):RETURN
770 A4=384:B4=RND(5):RETURN
780 REM ***       FINAL DE JOGADA       ***
790 PRINTà 152,"SEM MUNICAO !!";
800 IF SC>HI THEN HI=SC
810 PRINTà 408,"PLACAR:";SC;
820 PRINTà 472,"MELHOR PLACAR:";HI;
830 PRINTà 536,"ABATIDOS:";SH
840 PRINTà 792,"JOGA DE NOVO? (S/N)"
850 I$=INKEY$:IF I$<>"S" AND I$<>"N"
    THEN 850
860 IF I$="S" THEN SC=0:SH=0:GOTO 420
    ELSE CLS:END
870 REM ***   INTRODUZ ROTINA DE SOM   ***
880 S$="":FOR GH=1 TO 23:READ GJ:S$=S$+
    CHR$(GJ):NEXT GH
890 DATA 205,127,10,69,62,1,211,255,16,
    254,69,60,211,255,16,254,37,32,240,
    175,211,255,201
900 V=VARPTR(S$): J=PEEK(V+1)+256*
    PEEK(V+2)
910 IF PEEK(16396)=201 THEN POKE 16526,
    PEEK(V+1):POKE 16527,PEEK(V+2) ELSE
    DEFUSR=J-65536:RETURN
920 FOR X%=1 TO 15:L=USR(X%):NEXT
930 RETURN
```

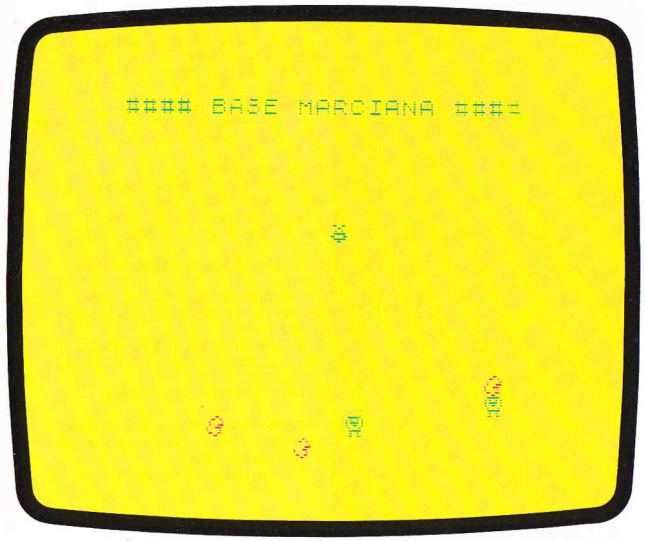
BASE MARCIANA

PARA TK 90X

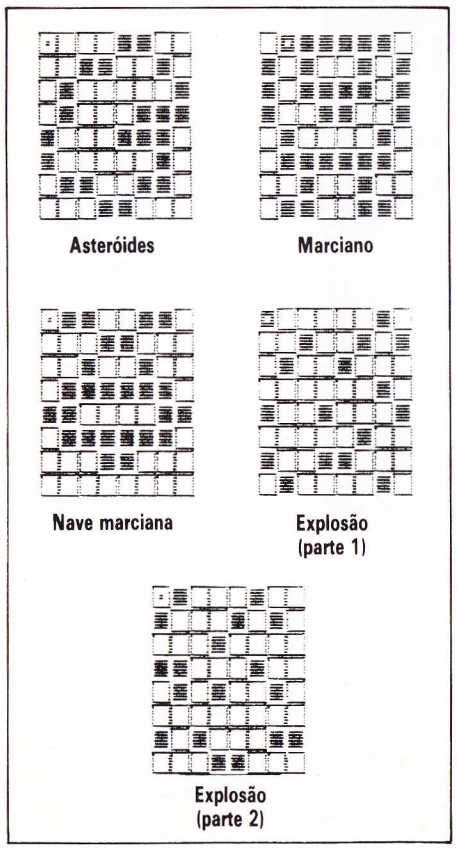
Você está nos controles de uma nave e sua missão é resgatar marcianos perdidos numa região repleta de asteróides. Cuidado! Não esbarre nos asteróides, pois isso provocará a explosão da nave. Após cumprir da melhor forma possível a tarefa, você será recebido com honras e homenagens. Para os movimentos, utilize os comandos: [5], nave para a esquerda; [6], para baixo; [7], para cima; e [8], para a direita. Cada resgate efetuado conta pontos.
Mas lembre-se: você tem pouco tempo para chegar à base. Antes de digitar o programa, veja na ilustração os caracteres a serem definidos.

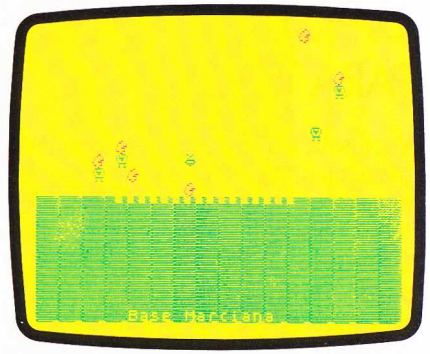
```
  1 REM Programa para o TK 90X
  2 REM
  3 REM      Base Marciana
  4 REM
  7 LET d=0
  8 INK 7: PAPER 0: BORDER 0: I
NVERSE 0: CLS
  9 LET x=0: LET y=0
 10 PRINT AT 8,4;"#### BASE MAR
CIANA ####"
 20 LET b=10: LET e=50+INT (RND
*300)
 30 LET c=16: LET c$="000008000
80005040200000808060811011"
 50 DIM a$(32*22)
 60 POKE 23692,0: LET a$=a$(33
TO )+"
                                "
 65 SOUND .09,VAL c$(1 TO 2): L
ET c$=c$(3 TO )+c$(1 TO 2)
 70 PRINT AT 21,31;" ": LET e=
e-1
 80 LET a=INT (RND*32): IF e<1
THEN PRINT AT 20,0;"                Base
Marciana                ": GO
 82 IF e<0 THEN PRINT AT 20,0;"
                                "
```

```
 84 IF e=y-18 THEN GOTO 272
 90 IF RND>.6 AND e>0 THEN PRIN
T AT 21,a;"@": LET a$(20*32+1+a)
="@"
100 LET d=d+.2
110 IF RND>.4 AND e>0 THEN PRIN
T AT 20,a;"@": LET a$(20*32+1+a)
="@"
120 IF a$(b*32+1+c)="@" THEN GO
TO 210
130 IF a$(b*32+1+c)="@" THEN LE
T d=d+1: BEEP .1,10
140 PRINT AT b,c;"&";AT b-1,c;"
";AT y,x;""
150 LET b$=INKEY$
155 BEEP .09,VAL c$(1 TO 2): L
ET c$=c$(3 TO )+c$(1 TO 2)
171 LET y=b-1
172 LET x=c
173 IF b$="" THEN GOTO 80
180 LET c=c+(b$="8")-(b$="5")-(
31 AND b$="8" AND c=30)+(31 AND
b$="5" AND c=1)
190 LET b=b+(b$="6")-(b$="7")-(
2 AND b$="6" AND b=20)+(21 AND
b$="7" AND b=1)
200 GOTO 80
210 PRINT AT b-1,c-1;"   ": FOR
a=1 TO 10
```



Besgate espacial
A tarefa mais difícil nesta aventura é se desviar dos numerosos asteróides. Um choque pode ser fatal.

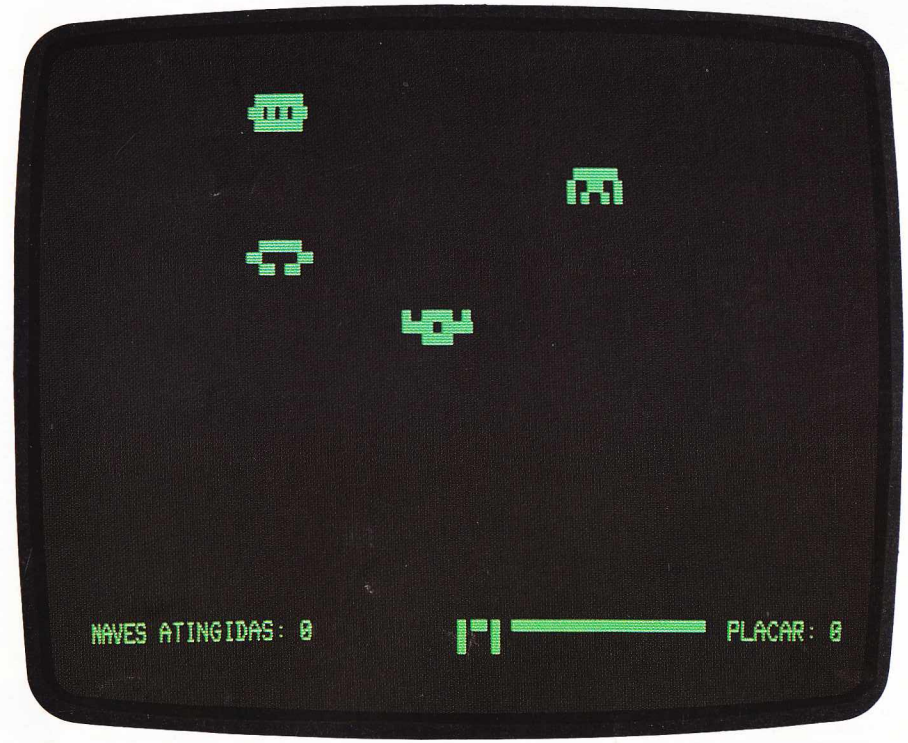





```
 220 PRINT AT b,c;"@"; FOR m=0 T
O 5: SOUND .00191,10: NEXT m: PR
INT AT b,c;"@"; FOR m=0 TO 5: SO
UND .00191,15: NEXT m
 230 NEXT a
 240 CLS
 250 PRINT AT 10,2;"Voce obteve
";INT (.5+10*d);" ponto(s).";AT
14,2;"Para novo jogo, digite 'J'
."
 260 PAUSE 4e4
 270 RUN
 272 LET z$="000004040400000404040
4040507090704040507090704000203000
0"
 273 FOR n=1 TO LEN z$ STEP 2
 274 SOUND .25,VAL z$(n TO n+1)
 276 NEXT n
 277 PAUSE 50
 278 IF INKEY$="" THEN GOTO 272
 280 CLS
 290 PRINT AT 10,2;"Voce obteve
";INT (.5+10*d);" ponto(s).";AT
14,2;"Para continuar o jogo, dig
ite 'J'."
 300 IF INKEY$<>"" THEN GOTO 300
 302 IF INKEY$<>"j" THEN GOTO 30
2
 310 GOTO 5
```


DISPARE
PARA LINHA TRS-80

Você dispõe de uma base lança-mísseis, mostrada na parte inferior da tela. Nesse jogo, seu objetivo é atingir as naves que passam sobre a base. A velocidade das naves é variável, mas nem por isso há vantagens em derrubar as lentas, porque as velozes valem mais pontos. Se você perceber que o tiro saiu errado e não atingirá a nave, aja com rapidez: o míssil, teleguiado, pode ser controlado durante o trajeto. Esse recurso lhe permite também derrubar mais de uma nave por disparo. Use a barra para disparar e as setas ou as teclas [Z] e [/] para direcionar os mísseis. O contador de naves atingidas e de pontos é sempre atualizado.




```
100 REM ***********************************
110 REM *          PARA LINHA TRS-80      *
120 REM *             DISPARE!            *
130 REM ***********************************
140 REM
150 REM ***   TELAS DE INSTRUCOES      ***
160 CLS:CLEAR 6000:PRINTà 21,
    "<<  DISPARE !  >>"
170 PRINT:PRINT:PRINT TAB(5) "O objetivo
    de <DISPARE>  e' abater o maior
    numero de ":PRINT TAB(5) "naves
    inimigas com  misseis teleguiaveis."
180 PRINT TAB(5) "Voce so' dispoe de 15
    misseis para abater o inimigo.":
    PRINT TAB(5) "A base dos misseis fica
    no centro inferior da tela.":
    PRINT TAB(5) "'A esquerda,  voce ve o
    numero de naves abatidas."
184 PRINTTAB(17) "USE QQ TECLA PARA CONTINUAR"
185 I$="":I$=INKEY$:IFI$=""THEN185
186 CLS
190 PRINT TAB(5) "'A direita fica o
    display do numero de pontos ganhos.":
    PRINT TAB(5) "Sempre existirao 4
```